
# ADDIS ABABA UNIVERSITY
# SCHOOL OF GRADUATE STUDIES
# COLLEGE OF NATURAL SCIECES
# DEPARTMENT OF COMPUTER SCIENCE

## Towards the Development of Stronger S-box

Haymanot Adane Yimam

**Advisor**: Dejene Ejigu (PhD)

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTERS OF SCIENCE IN COMPUTER SCIENCE

**ADDIS ABABA, ETHIOPIA**

***November, 2013***

**ADDIS ABABA UNIVERSITY**
**SCHOOL OF GRADUATE STUDIES**
**COLLEGE OF NATURAL SCIENCES**
**DEPARTMENT OF COMPUTER SCIENCE**

# Towards the Development of Stronger S-box

Haymanot Adane Yimam

**Advisor:** Dejene Ejigu (PhD)

**APPROVED BY**

**EXAMINING BOARD:**

1. Dr. Dejene Ejigu, Advisor _____
2. _____ _____
3. _____ _____

# Acknowledgements

First of all, I must thank **my God** for his gift and care that endowed me with the strength to finish my work. I must also thank **Saint Virgin Mary**, the mother of God, for her intercession and all the Saints in the Kingdom of God for their blessing. I would like to thank my thesis advisor **Dr. Dejene Ejigu** who gave me the opportunity to do this thesis, and also for the guidance and encouragement in making this work a reality. **Kassahun Desalegn,** thank you for your support with the mathematics used in this thesis. Finally I would also like to thank my parents and friends who helped me a lot in finishing this project within the limited time.

# Table of Contents

## List of Tables

## List of Figures

**Acronyms and Abbreviations**

AC: Autocorrelation

AES: Advanced Encryption Standard

ANF: Algebraic Normal Form

BIC: Bit Independence Criterion

DDT: Differential Distribution Table

LAT: Linear Approximation Table

MAC: Message Authentication Code

S-box: Substitution box

SAC: Strict Avalanche Criterion

WHT: Walsh Hadamard Transform

# Abstract

The heart of symmetric block cipher is the F function. This function relies on the use of the S-box. Therefore S-box should be constructed to be strong against cryptanalytic attack. One characteristics of strong S-box is nonlinearity that is the difficulty to approximate S-box by a set of linear equation. This thesis presents the design of stronger S-box using an optimized dynamic hill climbing method.

Boolean functions have application in a variety of systems; including block ciphers, stream ciphers and hash functions. In particular in block cipher it is applicable in S-box development. The strengths of both Boolean function and S-box are able to provide a cipher with strong properties to resist known and potential cryptanalytic attacks. Thus, in order to get a desirable measure we work on the cryptographic properties of Boolean function and S-box.

The main cryptographic properties required by strong Boolean functions and S-boxes are nonlinearity, autocorrelation, algebraic order, Strict Avalanche Criteria, Avalanche and Bit independency with different cryptographic applications requiring different acceptable measures of these and other properties. As combinations of cryptographic properties exhibited by functions can be conflicting, finding cryptographically strong functions often means that a trade-off needs to be made when optimizing property values.

This thesis focused on obtaining stronger S-box by customizing one of the heuristic Boolean function optimization method named as Dynamic Hill Climbing. Dynamic hill climbing method is efficient in optimizing one or more cryptographic properties of a single Boolean function, but strength of individual Boolean functions may not lead to strong S-box. Thus, we optimized the Dynamic Hill Climbing method to get both stronger Boolean functions for cryptographic application and S-box. The Optimized Dynamic Hill Climbing method and the overall value of Optimized Dynamic Hill Climbing method in optimizing S-box is clearly presented in this thesis.

Therefore, 15 S-boxes are generated using the customized Dynamic Hill Climbing algorithm to optimize the nonlinearity of S-box. The newly generated S-boxes are tested and compared with Advanced Encryption Standard, Camellia, Hierocrypt and Skipjack. As result, we have found S-boxes with nonlinearity 102.



**Keywords:** Boolean function, S-box, nonlinearity, algebraic order, autocorrelation.

# CHAPTER ONE - Introduction

## 1.1. Background

Cryptography is the science of securing stored data, as well as transmitted data through insecure networks using mathematics [48]. It is used for phone, fax and e-mail communication, bank transactions, bank account security, passwords and credit card transactions on the web.

Cryptography is one of the most important mechanisms used in the field of information security. Three main protections of data are offered by cryptography through the use of suitable and well-structured cryptographic cipher systems. These are Confidentiality, Integrity and Authentication.

Confidentiality is the act of keeping something private and secret from unauthorized disclosure. Confidentiality is important when there is a sensitive nature of network communication, such as trade secrets, client information subject to privacy laws or policies, or business strategies that depend on the element of surprise. It is achieved through strong encryption algorithms that can't be easily broken [45].

Integrity is provided by making sure the information has not been modified since creation or storage either maliciously or unintentionally [48]. If the data is in transfer the integrity service should assure that message is received without duplication, insertion, modification, reordering and replays. In cryptography integrity service can be achieved through different mechanisms such as, secure hash function and encipherment.

Authentication is the process of checking that the sender of the information is correctly identified and legitimate [1]. Symmetric encryption provides authentication among those who share the secret key. The two most commonly used techniques for message authentication are a message authentication code (MAC) and a digital signature.

By far important automated tool for protecting files and other information in network and communications is encryption. Two forms of encryption are in common use: conventional or symmetric encryption and public-key or asymmetric encryption [5].

In symmetric key algorithms (also called secret key, single key or one key algorithm), the encryption key and the decryption key are the same. It requires that the sender and the receiver agree on a key before they can communicate securely [7].

Symmetric algorithms can also be classified in to block cipher and stream cipher [7]. Block ciphers are algorithms that take part of or the whole blocks to be encrypted as input; whereas stream ciphers take a single byte or bit of the plaintext as input to the encryption algorithm.

Public key (asymmetric key) algorithms designed so that the key used for encryption is different from the key used for decryption; furthermore, the decryption key can't be calculated from the encryption key [7].

Boolean functions and substitution boxes are among the most common and essential components of ciphers. This is because they are able to provide a cipher with strengthening properties to resist known and potential cryptographic attacks [4]. Thus, it is not surprising that having significant amount of researches on Boolean functions and substitution boxes with optimal achievable measures of desirable cryptographic property.

S-box is a component of block encryption algorithm that obscures a fixed size data as block encryption does. They provide a means of substituting multiple bits (part of a whole blocks) of data for a completely different set of output bits. More importantly the use of strong S-boxes (those which possess good cryptographic properties) for substitution signify a complex relationship between input and output bits of the S-box.

An *nxm* substitution box (S-box) is a mapping from n input bits to m output bits, $S:\{0,1\}^n \rightarrow \{0,1\}^m$. For example for $n$=2 and $m$=2, there is a 2x2 S-box whose input and output bits are a 2 bit values. Thus, an S-box is simply a set of $m$ single output Boolean functions combined in a fixed order. There are $2^n$ inputs and $2^m$ possible outputs for an *nxm* S-box. Often considered as a look-up table, an *nxm* S-box, S, is commonly represented as a matrix of size $2^n \times m$, indexed as S[i] ($0 \leq i \leq 2^{n-1}$), each with $m$-bit entry [1].

An *nxm* S-box with $n = m$ (an equal number of input and output bits) may either contain distinct entries where each input is mapped to a distinct output or repeat S-box entries where multiple

inputs may be mapped to the same output and all possible outputs are not represented in the S-box. An *nxm* S-box which is both injective (one-to-one) and surjective (onto) is known as a bijective S-box. That is, each input maps to a distinct output entry and all possible outputs are present in the S-box. Bijective S-boxes may only exist when $n = m$ and are also called reversible since there must also exist a mapping from each distinct output entry to its corresponding input [2].

Mostly the security of block encryption algorithm depends on its S-box. Since S-box of block cipher is the only nonlinear component, its weakness can easily be recognized by linear and differential cryptanalysis. Therefore obtaining strong Boolean functions and S-boxes for incorporation into cryptographic cipher systems to enhance their security is ongoing research problem [1].

Therefore, the whole encryption algorithm will be strong if each components of the algorithm is strong. As a result, developing strong S-box will help to increase the security of the whole encryption algorithm.

An important application of Boolean functions in the field of cryptography is their use in substitution boxes (S-boxes). S-boxes perform a mapping from *n* binary inputs to *m* binary outputs and are used as a component in many block ciphers to assist in the provision of confusion(make the relationship between the key and the cipher text as complex as possible) during the encryption process [2]. S-boxes are therefore a key source of security for numerous block ciphers in current use. Therefore, the selection of Boolean functions with strong cryptographic properties reduces the effectiveness of advanced cryptographic attacks, including linear cryptanalysis and differential cryptanalysis on the S-box.

Three main classes of methods for generating S-boxes and Boolean functions with the desired cryptographic property are given in most publicly-available literatures as follows: Random searching, Construction methods and Evolutionary searching.

Among different S-box generation methods in this research we will focus on random S-box generation method in combination with Heuristic method of construction method. Heuristic techniques involve a process of iteratively improving a given function with respect to one or

more properties. This technique includes the genetic algorithm, hill climbing, simulated annealing and a combination of these [1].

## 1.2. Statement of the Problem

In the development of symmetric cryptosystem a significant portion of the time spent on design or analysis is centered on the substitution boxes (S-boxes) of the algorithm, this is because the only non linear component of the cipher is the S-box. The remaining parts of the algorithm can be written in the form of linear equation. Weaknesses in the S-box can therefore lead to a cryptosystem which is easily broken by using the most known cryptanalysis known as linear and differential cryptanalysis [3].

S-boxes generated with algebraic and heuristic methods which are incorporated in construction method are considered as cryptographically strong [1] [4]. But, S-boxes which are generated algebraically are exposed to algebraic attack [10]. However, the vast amount of experimentation so far performed using heuristic techniques has shown that the measured strength of the S-box generated with this technique is not at optimal level.

In this study, we will see each cryptographic property of Boolean function as well as S-box and investigate advantage and limitation of each method of S-box development, with the selected methodology among heuristic techniques; we will develop Boolean function with stronger cryptographic property. Finally we will try to generate cryptographically stronger S-box.

Accordingly, the problem to be investigated in this work includes:

- What are the components of S-box
- What are the properties of stronger S-box
- What S-boxes are in use today
- What are the methods to generate stronger S-box
- Can we have stronger S-box
- What are the testing criteria of S-box

## 1.3. Objectives

General objective of this study will be to generate cryptographically stronger S-box.

**Specific objectives**

The specific objectives of this research work will be:

- To identify good methodology among the existing heuristic methods that will help to generate Boolean function as well as S-box with better property.
- To apply the Heuristic techniques in the optimization of S-box properties.
- To measure the strength of S-box with respect to each property.
- To study and implement each property of S-box and their use on the security of the S-box.
- To generate a Boolean function that leads to stronger S-box.
- To contribute to the ongoing problem of obtaining stronger Boolean functions and S-boxes
- To increase the security of cryptographic cipher systems which utilize Boolean functions and S-box.

## 1.4. Scope of the Study

This research will focus on random S-box development method in combination with heuristic methods. There is other S-box construction method that focuses on Algebraic method, since the relationship between components in this S-box is exposed to algebraic attack, we will not use this method. Therefore our scope will be:

- Identifying best method of Boolean function and S-box optimization among heuristic optimization methods.
- Identifying properties of Boolean functions and S-box and their effect on S-box.
- Initial generation of S-box randomly having some property in mind.
- Optimizing the Boolean functions and S-box properties with heuristic methodology.
- Implement the identified property of S-box and compare security of our S-box with other S-boxes from literature.

## 1.5. Methodology

Construction of stronger S-box will have many components and developmental stages. On one hand there will be different tasks related to mathematics and on the other hand it will have different programming parts. The mathematical related task comprises of studying balanced property of Boolean function, non-linearity of Boolean function and S-box, Autocorrelation of S-box, Avalanche of S-box, Strict Avalanche Criterion (SAC) of S-box, Bit Independence Criterion (BIC), fixed point criterion and bijectivity criterion of S-box [1].

We will conduct literature review on strong S-box construction with heuristic methods, on the properties of Boolean functions and S-box. For mathematical knowledge acquiring, we will have continuous discussion with mathematicians in the domain.

Programming related task includes development of algorithms and implementation of different properties of the Boolean functions, the S-box and so on.

In researching for strong S-box development, C++ programming language will be employed as a major tool to implement the properties of Boolean functions and S-box. An open source random number generation system will be customized for initial S-box generation.

For the second step, which is optimization of Boolean function of randomly generated S-box, we will integrate the randomly generated S-box with the implementation of one of the selected heuristic method in order to improve its property. We will also collect previously constructed S-boxes by other researchers and compare them with our proposed S-boxes. Finally the strength of our new S-boxes will be evaluated against these S-boxes.

## 1.6. Application of the Result

After the development of S-box is successfully finished, the proposed S-box will be used for the following application.

- The strength of block encryption algorithm increases as the strength of the S-box increases. Therefore it is important to strengthen the whole encryption algorithm.
- By changing the standard algorithm's S-box, we can make the algorithms proprietary.

## 1.7. Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 presents literature review in which different concepts related to the thesis are presented. Chapter 3 is about related works which are done by other researchers. Chapter 4 deals with the design of our proposed S-box from the general architecture of our system presenting discussion of basic components and their interaction in the system. Chapter 5 is about implementation of the algorithms. It discusses the algorithms we used in achieving the goal of every component in the system. Chapter 6 deals with the experiments done and the results obtained compared with other S-boxes from literature. Chapter 7 finalizes our work by presenting a conclusion, future works and recommendation for the improvement of the system.

# CHAPTER TWO - Literature Review

Boolean function describes how to determine a Boolean value output based on some logical calculation from Boolean inputs. The properties of Boolean functions play a critical role in cryptography, particularly in the design of symmetric key algorithms.

In cryptography, an S-Box is a basic component of symmetric key algorithms which performs substitution. In block ciphers, they are typically used to obscure the relationship between the key and the cipher text. In general, an S-Box takes some number of input bits, *m*, and transforms them into some number of output bits, *n*, where *n* is not necessarily equal to *m*.

There are a number of methods to generate and optimize Boolean functions and S-boxes; heuristic is one among these methods. Heuristic techniques are driven by a direct search algorithm typically searching in a localized area from a specified starting point. Since heuristic techniques are not optimal, they are often applied to difficult combinatorial problems.

## 2.1. Boolean Functions

### 2.1.1. Overview of Boolean Function

The overall strength of a cryptographic cipher system (encryption algorithm) is dependent on the strength of the individual components, such as the permutation table, the substitution box (S-box), key scheduling algorithm etc [13]. A weakness in any of the individual components may lead to a catastrophic failure in the whole cipher.

Boolean functions have application in cryptography including in block ciphers, stream ciphers and hash functions. One of the most important applications of Boolean functions in the field of cryptography is their use in substitution boxes (S-boxes). S-boxes perform a mapping from *n* binary inputs to *m* binary outputs and are used as a component in many block ciphers to assist in the provision of confusion which is a technique that makes the relationship between the encryption key and cipher text as complex as possible during the encryption process. From all components of block encryption algorithm, S-box is the only nonlinear component meaning, there is no linear relationship between each of Boolean functions those build the S-box. Therefore S-boxes are a key source of security for numerous block ciphers in current use.

A Boolean function $f$ is a function whose domain is the vector space $F^n_2$ of binary $n$-tuples ($x_1$, $x_2$, . . . , $x_n$) that takes the values 0 and 1 and whose range is the set {0,1} [46]. Therefore, $f$ is a mapping from $n$ binary input ( $F^n_2$ ) into one binary output ( $F_2$ ). The set of Boolean functions on $F^n_2$ is denoted by $F_n$.

For instance, consider the exclusive-or function, defined by the following table:

Table 2. 1 Example of Boolean function with XOR operator

| $x_1$ | $x_2$ | $x_1 \oplus x_2$ |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

As can be shown from Table 2.1, the exclusive-or function can interpreted as a function $Z_2^2 \rightarrow Z_2$ that assigns (1, 1) → 0, (1, 0) → 1, (0, 1) → 1, (0, 0) → 0. It can also be written as a Boolean expression in the following way:

$$x_1 \oplus x_2 = (x_1 \oplus \neg x_2) \vee (\neg x_1 \oplus x_2)$$

Where $\oplus$ is a symbol to denote the bitwise excusive-or operator

We now present important representations for studying Boolean functions in the context of cryptography.

### 2.1.2. Characteristics of Boolean function

In this section we will discuss some important definitions and theorems fundamental to the cryptographic characteristics of Boolean functions. This includes specific Boolean function measures and a variety of representations used to express a Boolean function in the field of cryptography.

A Boolean function f(x): $Z_2^n \rightarrow Z_2$ such that x = ($x_1$ , $x_2$ ,… , $x_n$ ), is a mapping from $n$ binary inputs to one binary output. For $n$ variable there are $2^{2^n}$ possible Boolean functions each of $n$ bits [3]. We let $Z_2^n$ represents the set of all $2^{2^n}$ Boolean functions of $n$ variables. As can be seen

from Table 2.2 with *n*=3 input bits, we can have $2^8$ possible Boolean functions of 8-bit Boolean function with 2 possibilities for each bit.

The $2^8$ possible 8-bit Boolean functions are $f_1(x) = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$

$f_2(x) = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1$

$f_3(x) = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0$

. .

. .

. .

$f_{256}(x) = 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$

Table 2.2:

Example of Boolean function, n = 3.

Table 2. 2 Example of Boolean function, n=3

| $X_1$ | $X_2$ | $X_3$ | F(x) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

### 2.1.3. Representation of Boolean Function

Boolean functions can be represented using different forms, each with their own application in regard to cryptographic analysis.

**Truth Table**

The truth table of an *n*-variable Boolean function *f(x)* is a list of the output for each of the $2^n$ possible inputs to that Boolean function. There are two truth table representations each with their own application on cryptography namely binary truth table and polarity truth table.

Binary truth table is so named because the output symbols are elements of the set {0, 1}. Alternatively, for some applications it is useful to consider a Boolean function over the set {1, -1}. The truth table under this mapping is referred to as the polarity truth table [2].

Polarity truth table of an *n*-variable Boolean function is denoted by $\hat{f}(x)$. It is a vector containing $2^n$ elements, each element is in the set of {1,-1} [27]. The polarity truth table can be easily derived from the binary truth table and vice versa using the following mathematical relationship

$$\hat{f}(x) = 1 - 2f(x)$$

or a map of the Boolean function 0→1 and 1→ -1 [2]. An example of Truth table of Boolean function in binary and polarity form is given in Table 2.3.

The truth table representation of Boolean function is the basis [3] for the definition of several important cryptographic properties, from those properties we now discuss about Hamming weight, Hamming distance and Correlation.

Table 2. 3 Example of a Truth Table. N=3

| $X_1$ | $X_2$ | $X_3$ | f(x) | $\hat{f}(x)$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | -1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | -1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | -1 |
| 1 | 1 | 0 | 1 | -1 |
| 1 | 1 | 1 | 0 | 1 |

**Definition** 2.1 [2]: The Hamming weight of a Boolean function is defined as the number of 1s in the binary truth table, or the number of -1s in the polarity truth table (we shall use the notation wt(f) and $\mathrm{Wt}(\hat{f})$ interchangeably).

$$\mathrm{Wt}\,(f) = \sum_x f(x) \;=\; \frac{1}{2}\left(2^n - \sum_x \hat{f}(x)\right)$$

**Definition 2.2 [2]** : The Hamming distance between two functions f Є $Z_2^n$ and g Є $Z_2^n$ is defined as the number of truth table positions in which the functions differ and can be expressed as the Hamming weight of the XOR sum of two functions.

$$\mathrm{dist}\,(f, g) = \mathrm{wt}(f \oplus g)$$

The concept of correlation is significant [2] as it provides a convenient way to measure the degree of similarity between the two Boolean functions. The similarity can be expressed in terms of hamming distance between the two Boolean functions.

**Definition 2.3 [2]:** The correlation between two functions f Є $Z_2^n$ and g Є $Z_2^n$ is given by:

$$c(f,g) = 1 - \frac{dist(f,g)}{2^{n-1}}$$

Where dist (f,g) stands for the distance between the Boolean functions f and g.

Therefore, the result of the correlation is a rational number in the range [-1, 1]. From the definition we see that the upper bound of 1 is achieved when the Hamming distance between two functions is equal to zero. Similarly, the lower bound of -1 is achieved when the Hamming distance between two functions is equal to $2^n$. Correlation is an important tool in the analysis of pairs of functions particularly in relation to the concept of imbalance in a Boolean function.

A function is said to be balanced when half of the function values are equal to one [2];

$$\text{Wt(f)} = 2^{n-1} \text{ or alternatively, } \quad wt(\hat{f}) = 0$$

The imbalance of a Boolean function is defined to be [2]:

$$I(f) = | \text{ wt(f)} - 2^{n-1} | = 2^{n-1} ( C ( f(x), 0 ))$$

$$= 2^{n-1} (1 - \frac{dist(f(x),0)}{2^{n-1}} )$$

$$= 2^{n-1} - \text{dist}( f(x) , 0)$$

$$= |2^{n-1} - \text{wt (f)}|$$

Where *0* indicates the constant zero Boolean function and is therefore directly proportional to the magnitude of the correlation with the constant zero Boolean function, a function with zero imbalance is balanced and has no correlation to the constant functions.

**Algebraic Normal Form (ANF)**

The other important representation of a Boolean function is its algebraic normal form (ANF). Every ANF representation corresponds to a unique Boolean function truth table [2]. The ANF describes a Boolean function in terms of a unique XOR sum of AND products of the input variables [4].

**Definition 2.4 [2]:** The algebraic normal form (ANF) expresses a Boolean function as the XOR sum of ANDed input variables, such that given S = {1, 2,…, n} we can describe

$$f(x1,x2, \ldots, xn) = \bigvee_{I \subseteq S} aI \prod_{i \in I} Xi$$

Where V is bitwise exclusive-or operation, the coefficients $a_I$ € {0, 1} , form the elements of the truth table of the ANF of f(x).

When *n* is small, we can determine the ANF easily using a first principles approach [5]. Procedure of calculating ANF for *n=3* and example [2] is given in Table 2.4 and Table 2.5 respectively, which is continued from Table 2.3.

Table 2. 4 Calculating the ANF, n=3

| Input X | formula for $f(x)$ | solving for aI |
|---|---|---|
| 000 | $a0$ | $a0 = f(000)$ |
| 100 | $a0 + a1$ | $a1 = a0 + f(100)$ |
| 010 | $a0 + a2$ | $a2 = a0 + f(010)$ |
| 001 | $a0 + a3$ | $a3 = a0 + f(001)$ |
| 110 | $a0 + a1 + a2 + a12$ | $a12 = a0 + a1 + a2 + f(110)$ |
| 101 | $a0 + a1 + a3 + a13$ | $a13 = a0 + a1 + a3 + f(101)$ |
| 011 | $a0 + a2 + a3 + a23$ | $a23 = a0 + a2 + a3 + f(011)$ |
| 111 | $a0 + a1 + a2 + a3 + a123$ | $a123 = a0 + a1 + a2 + a3 + a12 + a13 + a23 + f(111)$ |

Table 2. 5 Example of an ANF, n=3

| X1 | X2 | X3 | f(x) | ANF |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |

The ANF of a Boolean function can also be derived from the binary truth table in a binary matrix transformation called the Algebraic Normal Form Transformation (ANFT). The ANFT is its own inverse and as such the binary truth table can be obtained from the ANF also using the ANFT.

**Definition 2.5** [4, 5] The ANF of a Boolean function f(x) is related to the TT by $ANF_f = A_n .f \bmod 2$, where the ANFT matrix, $A_n$ , of size $2^n * 2^n$ is defined recursively using the Kronecker product of matrices as follows

$$A_n = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes A_{n-1}, \text{ where } A_0 = [1]$$

$A_0 = [1]$

$$A_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes [1] = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

In general: $A_n = \begin{bmatrix} A_{n-1} & A_{n-1,0} \\ A_{n-1} & A_{n-1} \end{bmatrix}$

Where $A_{n-1,0}$ is $A_{n-1}$ with all entries zero value.

The algebraic normal form representation of Boolean function is used for the definition of Algebraic weight and Algebraic degree cryptographic properties [2]. We now discuss about these cryptographic properties.

The algebraic weight of a Boolean function is defined as the number of terms of the ANF. The algebraic weight denoted by awt(f) is defined as the number of coefficients $a_I$ in the algebraic normal form of f that are equal to one [2].

The algebraic order (algebraic degree), denoted ord(f) or deg(f), is defined as the number of variables in the largest product term in the Algebraic Normal Form of f, having a non-zero coefficient [28]. The algebraic degree of a Boolean function is a good indicator of the function's algebraic complexity. The higher the degree of a function, the greater is its algebraic complexity.

For example the ANF of input variable $x_1x_2x_3$ and $a_i$ € {0, 1} constant is:

Assume that all coefficients are 1.

$$f(x_1x_2x_3) = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_{12}x_1x_2 + a_{13}x_1x_3 + a_{23}x_2x_3 + a_{123}x_1x_2x_3$$

Therefore, awt(f) = 8 the number of non zero coefficients in an ANF of $f(a_0, a_1,a_2,a_3,a_{12},a_{13},a_{23},a_{123})$ and ord(f) = 3 the number of variables in the largest term in the ANF of $f(a_{123}x_1x_2x_3)$. The variables in this term are $x_1$, $x_2$ and $x_3$.

**Walsh Hadamard Transform (WHT)**

Walsh Hadamard Transform (WHT) is another way of representing Boolean function which represents information about the function from a significantly different frame of reference [3]. The WHT expresses a Boolean function in terms of its correlation with all linear functions and will be unique for each function [3].

**Definition 2.6 [2]** The Walsh Hadamard transform (WHT) of the polarity truth table of a Boolean function $\hat{f}(x)$, denoted by $\hat{F}(w)$ , is a measure of the correlation between a function and the set of all linear functions. It is defined by

$$\hat{F}(w) = \sum_x (-1)^{f(x)} (-1)^{lw(x)}$$

$$= \sum_x \hat{f}(x)\hat{l}w(x)$$

Table 2.6 shows the calculated WHT for the Boolean function f(x), n=3

Table 2. 6 Example of a WHT, n=3

| $X_1$ | $X_2$ | $X_3$ | f(x) | $\hat{f}(x)$ | $L_w(x)$ | $\hat{F}(w)$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | -1 | $X_3$ | 4 |
| 0 | 1 | 0 | 0 | 1 | $X_2$ | 0 |
| 0 | 1 | 1 | 1 | -1 | $X_2 + X_3$ | 4 |
| 1 | 0 | 0 | 0 | 1 | $X_1$ | 0 |
| 1 | 0 | 1 | 1 | -1 | $X_1 + X_3$ | 4 |
| 1 | 1 | 0 | 1 | -1 | $X_1 + X_2$ | 0 |
| 1 | 1 | 1 | 0 | 1 | $X_1 + X_2 + X_3$ | -4 |

As can be seen from Table 2.6 we see that $-2^n <= \hat{F}(w) <= 2^n$ for all w and each $\hat{F}(w)$ can be seen as directly proportional to the correlation with the corresponding linear function [2]. If $\hat{F}(0) = 0$, the function is balanced as well, the maximum absolute value of the WHT provides an important cryptographic measure of a Boolean function known as non-linearity [2] which is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function [2].

A linear function is defined as the XOR sum of single input variables. Similarly, the set of affine functions is defined as the set of linear functions and their complements.

**Definition 2.7 [2]**: A linear Boolean function, selected by $\omega \in Z^n_2$, is denoted by

$$L_w(x) = w_1x_1 \oplus w_2x_2 \oplus \ldots \oplus w_nx_n$$

where $w_i x_i$ denotes the bitwise AND of the $i^{th}$ bits of *w* and *x*, and $\oplus$ denotes bitwise XOR. $W = (w_1, w_2, \ldots w_n) \in Z^n_2$.

**Example:** for n=3,

$L_0(x)$, in this case w=000 and x Є $Z_2^n$ the value will be all zero because the value of w is zero.

$L_1(x)$, in this case w= 001 and x Є $Z_2^n$ the values are:

For x = 000, $L_w(x) = 0$ since 0.0^0.0^1.0 = 0

x = 001, $L_w(x) = 1$ since 0.0^0.0^1.1 = 1

x = 010, $L_w(x) = 0$ since 0.0^0.1^1.0 = 1

x = 011, $L_w(x) = 1$ since 0.0 ^0.1^1.1= 1

x = 100, $L_w(x) = 0$ since 0.1^0.0^1.0 = 0

x = 101, $L_w(x) = 1$ since 0.1^0.0^1.1 = 1

x = 110, $L_w(x) = 0$ since 0.1^0.1^1.0 = 0

x = 111, $L_w(x) = 1$ since 0.1^0.1^ 1.1 = 1

And continue for all possible w Є $Z_2^n$

**Definition 2.8 [2]:** The set of affine functions are the linear functions and their complements

$$A_{w,c}(x) = L_w(x) \oplus c$$

Where c Є {0, 1}

## Nonlinearity

One of important cryptographic properties of a Boolean function is its non-linearity [29]. The non-linearity of a Boolean function can be defined as the hamming distance between the Boolean function and the set of all affine functions.

**Definition 2.9 [11]** The nonlinearity of a Boolean function is defined as the minimum Hamming distance to the set of affine functions. Nonlinearity is given directly by observing$|\hat{F}\ max|$, the maximum absolute value occurring in$\hat{F}(w)$, and calculated as

$$N(f) = \frac{1}{2}\left(2^n - \left|\hat{F}max\right|\right)$$

Several criteria exist for measuring the nonlinearity of a Boolean function, including the minimum distance to any affine function and the order of a Boolean function [9]. All linear systems are easily breakable by linear cryptanalysis. It was found that the minimum distance to any affine function provides the most robust measure of nonlinearity, meaning that small changes to the truth table result in only small changes to this distance. Hence, the minimum distance to any affine function is used to define nonlinearity, such that the smaller the minimum distance to any affine function, the greater the nonlinearity.

**Example:** Measure of Non linearity for n=3 Boolean function. Let f(x) = 0 1 0 1 0 1 1 0 , n=3

$$N(f) = \frac{1}{2}\left(2^n - \left|\hat{F}max\right|\right)$$

$$\hat{F}(w) = \sum_x (-1)^{f(x)} (-1)^{lw(x)}$$

$$= \sum_x \hat{f}(x)\hat{l}w(x)$$

$(-1)^{f(x)}$ =1 -1 1 -1 1 -1 -1 1

$(-1)^{Lw(x)}$ = $l_{000}(x)$ = 0 0 0 0 0 0 0 0 → $(-1)^{Lw(x)}$ = 1 1 1 1 1 1 1 1

$(-1)^{f(x)}$ $(-1)^{Lw(x)}$ = 1 + (-1) + 1 + (-1) + 1 + (-1) + 1 + (-1) = 0

$L_{001}(x)$ = 0 1 0 1 0 1 0 1 $(-1)^{Lw(x)}$ = 1 -1 1 -1 1 -1 1 -1

$(-1)^{f(x)}$ $(-1)^{Lw(x)}$ = 1 + 1 +1 + 1 + 1 + 1 + 1 + 1 = 4

⋮

$L_{111}(x)$ = 0 1 1 0 1 0 0 1 $(-1)^{Lw(x)}$ = 1 -1 -1 1 -1 1 1 -1

$(-1)^{f(x)}$ $(-1)^{Lw(x)}$ = 1 + 1 + (-1) + (-1) + (-1) + (-1) + (-1) + (-1) = -4

$\left|\hat{F}max\right|$ = 4

Therefore: N (f) = $\frac{1}{2}(2^3 - 4)$

= 2

The autocorrelation function (AC) provides a useful description of a Boolean function in relation to its cryptographic properties. This property is derived from the derivative of a Boolean function. The derivative of a Boolean function shows the effect on the output of a function when there is a change in the input in the direction of s.

**Definition 2.10 [2]** The autocorrelation function, r(s), of a Boolean function f(x) is defined by the polarity truth table to be

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

Where x, s Є {0, 1,…,$2^n$-1}

The value of the autocorrelation function tells us the amount of the directional derivative for an input of a Boolean function shift in the direction of *s* over all *x* Є {0, 1,…,$2^n$-1}. From this definition of the autocorrelation function we note two important properties. Firstly, for every Boolean function *f(x), r(0)* = $2^n$, since $(\hat{f}(x))^2 = 1$ this implies that the function doesn't change or shift in the direction of s. Secondly, the value of r(s) when s ≠ 0 must be proportional to the correlation between *f(x)* and *f*(*x* ⊕s), i.e. r(s) = $2^n c(f(x), f(x \oplus x))$. As differential cryptanalysis [30] exploits the imbalance derivatives of Boolean function, AC is necessary in the analysis of Boolean functions for cryptography. Example of AC for the Boolean function given in table 2.3 by taking s as x is given in Table 2.7.

An important cryptographic measure observed directly from the AC function is the absolute indicator which is used to evaluate avalanche characteristics and defines the maximum absolute non-zero value of the AC function ($|AC|_{max}$) such that $0 \leq |r_{max}| \leq 2^n$. Absolute indicator can be derived from the autocorrelation function r(s).

Table 2. 7 Example of an AC, n=3

| $X_1$ | $X_2$ | $X_3$ | f(x) | $\hat{f}(x)$ | r(x) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 8 |
| 0 | 0 | 1 | 1 | -1 | -8 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | -1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | -1 | 0 |
| 1 | 1 | 0 | 1 | -1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |

### 2.1.4. Cryptographic properties of Boolean function

#### Avalanche Property

Avalanche effect is one of the important properties of Boolean function which was first proposed by Feistel [32]. The Boolean function *f* satisfies an avalanche criterion if and only if an average of one half of the output bits is changed whenever a single input bit is flipped [8].

Avalanche is the result of confusion and diffusion to the output [31]. Using this, the randomness property of the output can be achieved.

The derivative of a Boolean function gives a measure of the effect of the input bit difference on the output bits of the Boolean function. The equation for the derivative of a Boolean function is stated as follows.

$$d_s f(x) = f(x)f(x \oplus s)$$

**Definition 2.11 [2]** The avalanche effect with respect to a variable ei of a Boolean function f(x), denoted Aei(f), is defined as

$$Aei(f) = Prob(f(x) \oplus f(x \oplus ei) = 1)$$

as x ranges over all values.

## Completeness

Completeness is introduced by Kim and Davida [40] which says that each output bits of Boolean function is dependent on all of the input bits of the Boolean function. According to Kim and Davida," A Boolean function f has a good completeness effect when for each output bit i and each input bit j there exist two inputs which differ only in bit j but whose output bits differ in bit i." [8].

**Definition 2.12** [34] A function $Z_2^n \rightarrow Z_2^m$ is complete if and only if

$$\sum_{x \in Z_2^n} f(x) \oplus f(x \oplus C_i^n) > (0,0,\ldots,0)$$

For all i($1 \leq i \leq n$) where both the summation($\boldsymbol{\Sigma}$) and the greater-than(>) are component wise over $Z^m$.

## Strict Avalanche Criteria

The concept of SAC is initially introduced by Webster and Tavares [51] by combining the property of Avalanche and Completeness of Boolean function. SAC is needed whenever the cryptographer needs some "complex" mapping *f* of *n* bits on to one bit.

SAC says that the probability of changing the output bits is half whenever a single input bit is flipped. Therefore, the desired probability value for SAC test is half [12].

**Definition 2.13 [2]** A Boolean function f of n variables satisfies SAC if for each unit vector $e_k$,

$1 \leq k \leq n$, $wt((\Delta f)\, e_k) = 2^{n-1}$.

For $1 \leq m \leq n-2$, if f satisfies SAC (m-1) then it satisfies SAC (m) if and only if any function of n-m variables obtained from f by keeping m input bits constant satisfies SAC.

## Correlation immunity and resilience

The correlation immunity of a Boolean function is a measure of the degree to the independence between linear combination of the input bits and the output bits [3]. Specifically, a Boolean function is said to be correlation-immune of order *m* if every subset of *m* or fewer variables in $x_1,x_2,\ldots,x_n$ is statistically independent of the value of $f(x_1,x_2,\ldots,x_n)$ [2].

An *n*-variable Boolean function, f(x), is $m^{th}$-order correlation immune, denoted CI (m), if it is statistically independent of the subset of *m* input variables where $1 \leq m \leq N$.

**Definition 2.14** [2] Let f(x) be a Boolean function of *n*-variables with polarity WHT $\hat{F}(w)$, the function will have correlation immunity of order *m* if and only if $\hat{F}(w)= 0$ for all non-zero *w* with $wt\ (w) \leq m$. In cryptography it is necessary to have correlation immune Boolean function in order to resist against divide and conquer attack [10]. A function which is both correlation immune and balanced is known as Resilience.

**Definition 2.15 [**2] Let *f(x)* be a Boolean function of *n* variables with polarity WHT$\hat{F}(w)$. The function can be described as being *m*-resilient if and only if $\hat{F}(w)= 0$ for all *w* with $wt(w) \leq m$.

## 2.1.5. Bent Boolean Function

Bent Function is a Boolean function that has optimal distance from linear function, as a result bent function is a function with a perfect nonlinearity [9].

Boolean functions with odd number of variables can't be bent. Only functions that has even number of variables satisfy the characteristics of bent [35]. Since the Walsh value indicates the correlation of the function with the corresponding linear function, the walsh hadmard transform of a bent function consists of entirely two values that is $+2^{n/2}$ or $-2^{n/2}$ [42]; Thus the non-linearity of the bent function is calculated as

$$\frac{2^n - 2^{n/2}}{2}$$

This is the maximum non-linearity for a Boolean function with *n* number of variables (n is even) [7]. This indicates that the bent function is at maximum distance from the set of all affine functions as well as from linear structures.

From the cryptographic point of view, it is important to know that even though bent functions have the highest non-linearity, they are necessarily non-balanced [36].

## 2.2. **The S-box**

### 2.2.1. **Definition of S-box**

An *nxm* S-box is a non-linear transformation[37] from *n* input bits to *m* output bits, $S:\{0,1\}^n \rightarrow \{0,1\}^m$ [3]. Thus, an S-box is simply a set of *m* single output Boolean functions combined in a fixed order. There are $2^n$ inputs and $2^m$ possible outputs for an *nxm* S-box. Often considered as a look-up table, an *nxm* S-box, *S*, is commonly represented as a matrix of size $2^n$ *x m*, indexed as S[i] ($0 \leq i \leq 2^{n-1}$), each an *m*-bit entry.

The arrangement [41] of Boolean functions into output bits is as follows. Let the *n* functions be $f_1, f_2 \ldots . f_n$, where each function $f_i$ corresponds to a binary vector $f_i$ of length $2^n$. Then the S-box $S = [f_1, f_2 \ldots . . f_n]$ is a $2^n$ *x n* bit matrix with the $f_i$ as column vectors. Any given input vector $x = x_1, x_2 \ldots x_n$, maps to an output vector $y = y_1, y_2 \ldots . . y_n$, by the assignment $yi = f_i (x_1, x_2 \ldots . . x_n)$.

As a simple example, let n = 3 and $f_1 = (00001111)$, $f_2 = (00110011)$, $f_3 = (01010101)$. Then

$$S = S^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}^t$$

In this case, the S-box is a non-permuted mapping of the integers 0-7 onto themselves.
Based on the number of input and output bits of S-box, we can classify S-box as one-to-one, onto and bijective. The definition for each classification is stated as follows.

An *nxm* S-box with n < m (a greater number of output bits than input bits) is incapable of having all possible outputs as entries in the S-box. If all output entries are distinct then the S-box is said to be injective (one-to-one).

For an *nxm* S-box with n > m (a greater number of input bits than output bits) there must exist repeat S-box entries. If all possible outputs are present in the S-box then the S-box is said to be subjective (onto).

An *nxm* S-box with n = m (an equal number of input and output bits) may either contain distinct entries where each input is mapped to a distinct output OR repeat S-box entries where multiple inputs may be mapped to the same output and all possible outputs are not represented in the S-box [39]. An *nxm* S-box which is both injective and subjective is known as a bijective S-box. That is, each input maps to a distinct output entry and all possible outputs are present in the S-box. Bijective S-boxes may only exist when n = m and are also called reversible since there must also exist a mapping from each distinct output entry to its corresponding input.

## 2.2.2. Cryptographic properties of S-box

### Nonlinearity of S-box

The non-linearity of an S-box *S* is $nl(S) = min\ nl(f), f \in C$, where *C* is the set of all nontrivial linear combinations of the columns of *S*. In other words, *NL (S)* equals the minimum Hamming distance between all the component functions of *S* and all affine functions on *n* variables. This generalization is closely related to the linear attack because; the linear attack finds the opportunity to approximate the S-box by a set of linear equation.

The nonlinearity of S-boxes is clearly a right and left affine invariant and the nonlinearity of an S-box *F* does not change if we add to *F* an affine function. Moreover, if *A* is a surjective linear (or affine) function from $F^p_2$ (where *p* is some positive integer) into $F^n_2$, then it is easily shown that $NL(F \circ A) = 2^{p-n} NL(F)$.

According to the equality relating the nonlinearity of a Boolean function to the maximum magnitude of its Walsh transform, we have:

$$NL(F) = 2^{n-1} - \frac{1}{2} \max_{v \in F_2^{m^*}; u \in F_2^n} \left| \sum_{x \in F_2^n} (-1)^{v.F(x) \oplus u.x} \right|$$

Note that " $\max_{v \in F_2^{m^*}; u \in F_2^n}$ " can be replaced by " $\max_{(u,v) \in F_2^n X\, F_2^m; (u,v) \neq (0,0)}$ ". Hence, if n=m and if F is a permutation, then F and its inverse $F^{-1}$ have the same nonlinearity (change the variable x into $F^{-1}(x)$.

## Algebraic Degree of an S-box

The algebraic degree of an S-box (and similarly a Boolean function) is desired to be as high as possible in order to resist a cryptanalytic attack known as low order approximation. The measure of S-box degree is defined below.

**Definition 2.16 [3]** let S = ($f_1$, $f_2$,…, $f_m$) be an n x m S-box where $f_i$ (i=1,…,m) are *n* variable Boolean functions. Let $g_j$ be the set of linear combinations of $f_i$ (i=1,…,m) (which include the function $f_i$). Then the algebraic degree of *S*, denoted by deg(Sn,m), is defined as

$$Deg(Sn,m) = \min_g\{deg(g_j)\}\ (j=1,\ldots,2^m-1)$$

## Autocorrelation of S-box

The Autocorrelation of an S-box is the maximum autocorrelation of all linear combination of each Boolean function of an S-box.

Let S=($f_1$,$f_2$,…,$f_m$) be an *n x m* S-box where $f_i$ (i=1,…,m) are *n* variable Boolean functions. Let $A_i$ be the set of linear combinations of each Boolean function of an S-box. Then the autocorrelation of an S-box [14] denoted by AC(Sn,m), is defined as

$$AC(Sn,m) = Max_A\{r(A_i)\}\ (i=1,\ldots,2^m-1)$$

**Definition 2.17 [12]** A function $Z_2^n \rightarrow Z_2^m$ is complete if and only if

$$\sum_{x \in Z_2^n} f(x) \oplus f(x \oplus C_i^n) > (0,0,\ldots,0)$$

For all i($1 \leq i \leq n$) where both the summation($\boldsymbol{\Sigma}$) and the greater-than(>) are component wise over $Z^m$. $C_i^n$ means an *n* dimensional vector with Hamming Weight 1 at the $i^{th}$ position.

This means that each output bit depends on all of the input bits. Thus, if it is possible to find the simplest Boolean expression for each output bit in terms of the input bit, each of these expressions would have to contain all of the input bits if the function is complete.

**Definition 2.18 [12]** (**Avalanche effect**).A function $Z_2^n \rightarrow Z_2^m$ exhibits the avalanche effect if and only if

$$\sum_{x \in Z_2^n} wt(f(x) \oplus f(x \oplus C_i^n)) = m2^{n-1}$$

For all i ($1 \leq i \leq n$) ,where $C_i^n$ means an *n* dimensional vector with Hamming Weight 1 at the $i^{th}$ position. This means that an average of one half of the output bits change whenever a single input bit is complemented.

**Definition 2.19 [12] (Strict Avalanche Criterion, strong S-box**). We say that a function f: $Z_2^n \rightarrow Z_2^n$ satisfies the SAC, or f is a strong S-box, if for all i($1 \leq i \leq n$) there hold the following equations:

$$\sum_{x \in Z_2^n} f(x) \oplus f(x \oplus C_i^n) = (2^{n-1}, 2^{n-1}, \ldots, 2^{n-1}).$$

In particular, if f: $Z_2^n \longrightarrow Z_2^m$ satisfies the SAC, f is called a Boolean strong S-box.

If a function satisfies the SAC, each of its output bits should change with a probability of one half whenever a single input bit is complemented. Clearly, a strong S-box is complete and exhibits the avalanche effect.

If some output bits depend on only a few input bits, then, by observing a significant number of input-output pairs such as chosen plaintext attack, a cryptanalyst might be able to detect these relations and use this information to aid in the search for the key.

**Definition 2.20 [33]: (1st order SAC)** A function f: $Z_2^n \rightarrow Z_2^m$ is said to satisfy the 1-st order SAC if and only if

- f satisfies the SAC, and
- Every function obtained from f by keeping the i-th input bit constant and equal to c satisfies the SAC as well for every i ∈ {1,2,…,n}, and for c=0 and c=1.

Naturally, the SAC defined in **definition 2.19** can be said to be the 0-th order SAC too. To verify whether an *n*-bit input Boolean function satisfies the 1-st order SAC or not, at most n + n(n-1) checks are required. n checks correspond the 0-th order SAC and n(n-1) checks corresponding the 1-st order SAC. This definition can be extended to the k-th order SAC where $1 \leq k \leq n-2$ if k input bits of f(x) are kept constant.

**Definition 2.21 [33]:** (**k-th order SAC**). A function f: $Z_2^n \rightarrow Z_2^m$ is said to satisfy the k-th order SAC if and only if

- f satisfies the (k-1)-th SAC, and
- any function obtained from f by keeping k of its input bits constant satisfies the SAC as well (this must be true for any choice of the positions and of the values of k constant bits).

Therefore verifying whether an n-bit input Boolean function satisfying the k-th order SAC or not required at most $n + n(n-1) + \binom{n}{2}(n-2) + \ldots + \binom{n}{k}(n-k)$ checks.

Afterwards, when we talk of a function satisfying the SAC without specifying the order, the function at least satisfies the 0-th order SAC. Moreover, if an *n*-bit input Boolean function satisfies the *(n-2)*-th order SAC, the function is referred to as a Boolean function satisfying the maximum order SAC in the sense that the *(n-2)*-th order SAC is maximally achievable in Boolean functions.

**Bit Independence Criterion (BIC)**

A function f : $\{0,1\}^n \rightarrow \{0,1\}^n$ satisfies the BIC if for all i, j, k ε {1,2,…,n} with j ≠ k, inverting input bit *i* causes output bits *j* and *k* to change independently. To measure the bit independence concept, one needs the correlation coefficient between the *j* 'th and k 'th components of the output difference string, which is called the avalanche vector $A^{ei}$

A bit independence parameter corresponding to the effect of the i 'th input bit change on the j 'th and k'th bits of $A^{ei}$ is defined as [43]

$$BIC(a_j, a_k) = \max_{1 \le i \le n} \left| corr\left(a_j^{ei}, a_k^{ei\prime}\right) \right|$$

Overall, the Bit Independence Criterion (BIC) parameter for the S-box function f is then found as

$$BIC(f) = \max_{\substack{1 \le j,k \le n \\ j \ne k}} BIC\left(a_j, a_k\right)$$

This demonstrates how close f is to satisfying the BIC. BIC (f***)*** takes values in [0, 1]. It is ideally equal to 0 and, in the worst case, it is equal to1.

## 2.3. Heuristic Techniques

### 2.3.1. Introduction to Heuristic Technique

Heuristic method is a search method to optimize Boolean function in order to develop stronger S-box. Heuristic techniques involve a process of iteratively improving a given function with respect to one or more properties. Therefore, it involves algorithms that either give nearly the right answers or provide a solution not for all instances of the problem. Heuristic technique involves [2] genetic algorithm, hill climbing and simulated annealing etc.

**Genetic algorithm**: work with a population of candidate functions. They involve the application of three fundamental operations that are inspired by natural evolution (selection, crossover and mutation), with the aim of producing future populations containing functions with the desired properties.

**Hill climbing:** optimize the cryptographic properties of a given randomly generated Boolean function [2], [3] as well as S-box [1] with the basic idea of searching, at each iteration, for elements of a function to modify which will result in an improvement in the results already obtained. At the end of the process, it is expected that the final output will represent the best solution obtainable.

**Simulated annealing:** is a generic probabilistic meta heuristic for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities) [14].
For the purpose of this paper we will see the detail of hill climbing technique.

Since the late 1990s hill climbing techniques have been demonstrated to be effective in research of cryptography. Hill climbing is the process whereby one or more distinct elements in the truth table of a function are complemented in order to make iterative improvements to the cryptographic properties or fitness of the function. Fitness function is referred to as the measure of a particular cryptographic property or properties of a given Boolean function.

According to Millan, Clark and Dawson in [44], fitness function is categorized as weak acceptance and strong acceptance. Weak acceptance will accept an incremental change in the truth table even if such a change produces no increase in the fitness of the new function, provided that there is no decrease in the fitness. On the other hand, strong acceptance condition will only accept an incremental change in the truth table when such a change produces an increase in the fitness of the new function. Thus, the only time an increase in the fitness is forced is when a strong acceptance condition is imposed.

According to [3] there are two hill climbing methods namely Traditional and Dynamic hill climbing. The detail of each method is given as follows.

### 2.3.2. Traditional Hill Climbing

This method is based on the effect of small truth table change on the WHT and nonlinearity. The change can be one step (one bit change) or two step (two bits change) [2].
Any one step truth table change causes

$$\Delta\hat{F}(w) \in \{-2 , 2\} \text{ for all } w$$

and hence

$$\Delta N_f(w) \in \{-1 , 1\} ,$$

And any two step changes causes

$$\Delta\hat{F}(w) \in \{-4 , 0 , 4\} \text{ for all } w$$

and hence

$$\Delta N_f(w) \in \{-2 , 0, 2\}.$$

If our fitness function is nonlinearity, appropriately choosing the values in the truth table provides a means in an improvement of the nonlinearity of the function. In the case of balanced Boolean function, in order to maintain the balanced property of the Boolean function, we should use a two step change by selecting two points in the function which have different functional values i.e. the complement of these values maintain the hamming weight of the function. The key steps for the traditional hill climbing are listed in Algorithm 2.1.

**Algorithm 2.1**

1. *Define fit(f) as the fitness function of a Boolean function, f.*
2. *Generate a random N-variable Boolean function, f(x), x = 0,..,N - 1.*
3. *Iterate:*

    a) *Form improvement sets I(f) and compute fit(f).*
    b) *Select two bit positions, i and j, where f(i) ≠ f(j).*
    c) *Derive candidate function g(x) = f(x), where x = 0, …, i-1, i+1, … , j- 1, j + 1, … ,$2^N$ - 1; and g(i) = f(i) ⊕1, g(j) = f(j) ⊕1.*
    d) *Form improvement sets I(g) and compute fit(g). If* $\forall \emptyset \in I(g), |WHT(\emptyset)| < WHTmax$ *, then let* $f = g$.
    e) *If loop reaches limit of iterations with no further improvements, exit.*
    f) *Output final Boolean function, g, representing best achievable fitness.*
4. *Repeat 2. and 3. as required, to hill climb desired number of Boolean functions.*

Figure 2. 1 Traditional Hill Climbing Algorithm

In this method, every pair of points in the truth table can be categorized as improvement set, static set and reduction set. The values are categorized in the following manner.
The improvement set is defined as the set of all pairs ($x_1$ , $x_2$) for which complementing the corresponding function values produces $\Delta|\hat{F}_{\max}| = -4$ and thus an increase to nonlinearity.

The static set is defined as the set of all pairs ($x_1$ , $x_2$) for which complementing the corresponding function values produces $\Delta|\hat{F}_{\max}| = 0$ and thus no change to nonlinearity.

Finally, the reduction set is defined as the set of all pairs ($x_1$ , $x_2$) for which the complementation of the corresponding function values produces $\Delta|\hat{F}_{\max}| = 4$ and thus a decrease to nonlinearity.

Formally, the classification of a pair ($x_1$ , $x_2$), such that $f(x_1) \neq f(x_2)$, to one of the three change sets can be given specifically in terms of those WHT positions near and equal to $\Delta|\hat{F}_{max}|$ as follows.

**Improvement Set** The set of all pairs ($x_1$; $x_2$) that satisfy all of the following conditions:

1. $\Delta\hat{F} = -4$ for all $w \in \{ w: \hat{F}(w) = |\hat{F}_{max}|\}$
2. $\Delta\hat{F} = +4$ for all $w \in \{ w: \hat{F}(w) = -|\hat{F}_{max}|\}$
3. $\Delta\hat{F} \neq +4$ for all $w \in \{ w: \hat{F}(w) = |\hat{F}_{max}| - 4\}$
4. $\Delta\hat{F} \neq -4$ for all $w \in \{ w: \hat{F}(w) = |\hat{F}_{max}| + 4\}$

**Static Set** The set of all pairs ($x1$; $x2$) that satisfy all of the following conditions:

1. $\Delta\hat{F} \neq +4$ for all $w \in \{ w: \hat{F}(w) = |\hat{F}_{max}|\}$
2. $\Delta\hat{F} \neq -4$ for all $w \in \{ w: \hat{F}(w) = -|\hat{F}_{max}|\}$
3. Either $\Delta\hat{F} = 0$ for at least one $w \in \{w : \hat{F}(w) = \pm |\hat{F}_{max}|\}$
   Or $\Delta\hat{F} = +4$ for at least one $w \in \{ w: \hat{F}(w) = |\hat{F}_{max}| - 4 \}$
   Or $\Delta\hat{F} = -4$ for at least one $w \in \{w: \hat{F}(w) = -|\hat{F}_{max}| + 4\}$

**Reduction Set** The set of all pairs ($x_1$; $x_2$) that satisfy all of the following condition:

1. Either $\Delta\hat{F} = +4$ for at least one $w \in \{w: \hat{F}(w) = |\hat{F}_{max}|\}$
   Or $\Delta\hat{F} = -4$ for at least one $w \in \{ w: \hat{F}(w) = -|\hat{F}_{max}|\}$

Basically there are two methods of traditional hill climbing namely strong and weak hill climbing. Strong hill climbing algorithm focuses on the improvement set of the function and the algorithm terminates when the improvement set of the current function is empty or when the maximum limit of the iteration have been implemented.

In the case of weak hill climbing the algorithm focuses not only on the improvement set it will search through static set if the improvement set is empty. The static set of the current function doesn't increase the nonlinearity of the function but it will help to get other function with the same nonlinearity and that has improvement set.

### 2.3.3. Dynamic Hill Climbing

The major limitation of strong hill climbing is that since it doesn't go through the static set, it terminates at the local maximum. The weak hill climbing was able to improve somewhat on this concept, allowing ridge points to be traversed in any iteration where the change implemented was such that nonlinearity remained constant. The weak improvement set is the combined set of improvement and static set; because of the existence of small number of improvement set in the combined set, the probability of selecting improvement set from this combination of improvement and static set is low.

The dynamic version of hill climbing goes one step further in providing for the improvement of nonlinearity by continually checking for the existence of an improvement set pair at every iteration. The basic idea of this approach is implement either strong or weak hill climbing depending on the classification of the function in a given iteration.

The implementation of Dynamic hill climbing involves the use of improvement set at any iteration until the improvement set becomes empty. When the improvement set becomes empty, the static set becomes implemented to traverse the ridge. Iteratively the static set is implemented until the nonempty improvement set function is generated in which case the improvement set can be again utilized to improve the nonlinearity, or until the static set becomes empty.

Fuller on [2] suggested that when the current function forms either a ridge or plateau, a member of the static set is selected for modification of the function, while when the function is at a slope; change-point, valley or ditch, a member of the improvement set is selected for modification. It is important to note that this type of hill climbing may never terminate, but simply move continuously along ridges in the terrain. To prevent this occurrence, it is vital to use a static-limit parameter to specify the maximum number of non-increasing nonlinearity function changes that occur.

Algorithm 2.2 is a dynamic hill climbing algorithm with random number generation which is used to generate random Boolean function with the desired hamming weight to be improved.

***Algorithm 2.2***

***Input*** *- n; input size*

*- static_limit; maximum static movements to occur concurrently*

*- hwt; the Hamming weight of the desired function*

***Output*** *- f(x); a function of high nonlinearity*

1. *Generate of random function f(x) of input size n and Hamming weight hwt, and find its corresponding WHT.*
2. *Find the 2-step improvement set of f(x), such that* $f(x_1) \neq f(x_2)$ *for all pairs* $(x_1, x_2)$.
3. *While the improvement set is not empty:*
    a) *Select a random pair* $(x_1, x_2)$ *from the improvement set and complement* $f(x_1)$ *and* $f(x_2)$.
    b) *Find the new WHT of f(x).*
    c) *Find the new 2-step improvement set.*
4. *Set the variable static to zero.*
5. *Find the 2-step static set of f(x), such that* $f(x_1) \neq f(x_2)$ *for all pairs* $(x_1, x_2)$.
6. *If the static set is not empty and static does not exceed static_limit:*
    a) *Increment static.*
    b) *Select a random pair* $(x_1, x_2)$ *from the static set and complement* $f(x_1)$ *and* $f(x_2)$.
    c) *Find the new WHT of f(x).*
    d) *Find the new 2-step improvement set.*
    e) *If the improvement set is not empty return to step 3, else return to step 5.*
7. *Return f(x).*

Figure 2. 2 Dynamic Hill Climbing Algorithm

### 2.3.4. Classification of function

According to Fuller in [2], based on the three change sets (improvement, static and reduction) each function is classified in the Boolean terrain to distinguish between functions on the basis of the existence of the change sets.

"The existence of each of the three change sets for a given function provides the means for describing the "shape" of the Boolean terrain in the search space surrounding that function" [2]. Therefore by viewing functions that result from a 2-step change as surrounding the original function, this system of classification provides a picture of the shape of the Boolean terrain with respect to nonlinearity. The result of the analysis from paper [2] is reported on table 2.1.

Table 2.1 shows that when the function is at slope level all the three change sets are not empty that means using one pair of the improvement set the nonlinearity of the function is ready to increase and by complementing pair of bits from static set, the function can be changed to another function that may or may not have an improvement state, whereas at the peak level the only nonempty set is the reduction set this indicates that the nonlinearity of the functions in this category is at maximum limit.

Table 2. 8 Classification of Functions in the Boolean Terrain

| **Function classification** | **Size of the improvement set** | **Size of the static set** | **Size of the reduction set** |
|---|---|---|---|
| Peak | 0 | 0 | $\geq 1$ |
| Plateau | 0 | $\geq 1$ | 0 |
| Ridge | 0 | $\geq 1$ | $\geq 1$ |
| Trough | $\geq 1$ | 0 | 0 |
| Saddle Point | $\geq 1$ | 0 | $\geq 1$ |
| Valley | $\geq 1$ | $\geq 1$ | 0 |
| Slope | $\geq 1$ | $\geq 1$ | $\geq 1$ |

# CHAPTER THREE - Related Works

This chapter presents review of related works which were done on development of S-box and Boolean function. The papers adopted development techniques that include Algebraic construction methods [13, 18, 20, and 22], Heuristic development methods [14, 15, 17], Evolutionary method [16], Random [19] and we will see paper with new techniques of S-box development [21]. Each paper proposed S-box with respect to one or more cryptographic properties.

## 3.1. Algebraically Designed S-boxes

On [13, 18, 20, 22], these papers proposed to solve the problem of getting cryptographically strong S-boxes using algebraic technique.

Since many researchers are paying attention to improve the quality of S-box design, there are significantly many methods of algebraically generating S-box. In this category even if the papers use mathematical formula to generate the S-box, the way they use the formula and the formula itself is different. On [13], the researchers use *(17z+13)/(29z+41)* fractional linear transformation which is one of an action of projective general linear group PGL (2, GF ($2^8$)) applied on Galois field GF($2^8$), where z ∈ GF($2^8$).

In the same way Hussain and others on paper [22] use the same algebraic method as [13] the only difference is the fractional linear transformation. According to the proposed algorithm, The linear fractional transformation used in the construction of S-boxes *( f (z) = (( 35z + 15) / (9z + 5)))* is also formed with the action of projective linear group PGL(2, GF($2^8$)) on GF($2^8$). The calculated values of *f(z)* are replaced with their binary value equivalent, represented as some power of *w*, where *w* is the root of the primitive irreducible polynomial, $P(x) = X^8 + X^4 + X^3 + X^2 + X$: The resulting values from GF($2^8$) are then solved to determine the eight-bit binary value to be used in S-box. However, the average non-linearity of the proposed S-box is 104 and 106 respectively.

In [18], the researchers focused on S-box of Rijndeal algorithm. Since the algorithm requires look up operation for the S-box at every round, the S-Box computation is the most time-consuming operation and storing the S-box on read only memory causes security problem. To avoid such vulnerability, the researchers on this paper computed the S-Box values on a real-time basis. However, using the Galois Fields (GF) (which is originally proposed by Rijndael) renders this option undesirable. To overcome these limitations, real time computations of these inverses can be performed using arithmetic modulo some number, as the researchers said modulo arithmetic approach is efficient and takes significantly less time and space compared to the GF. They also proposed that legitimate candidates for modulo arithmetic are numbers that are powers of two and prime numbers. Henceforth, arithmetic modulo a power of two will be referred to as *AM* ($2^n$), and arithmetic modulo a prime will be referred to as *AM (P).*

In other way of Algebraic construction of S-box on paper [20] the researchers proposed two construction method of nonlinear resilient S-box with a given degree. The first method is to construct highly nonlinear *(n,m,t)* resilient S-box (S-box that has n-bit input, m bit output and t-resilient) given the parameters n, m, t and d. The researchers use the concept of linear code to construct these S-boxes. First they construct *(d+1 , m)* S-box called *T(x)* then Obtain an *[n − d − 1,m, t + 1]* linear code C with basis $\{c_1, \dots, c_m\}$, construct an *(n − d − 1,m, t)* resilient S-box $H(y) = (c_1 \cdot y, \dots, c_m \cdot y)$. Finally let $G_1$: $F^m{}_2 \rightarrow F^m{}_2$ be obtained from the inverse mapping, then output an S-box $F(x, y) = T(x) + G_1 \circ H(y)$. With this construction method they can develop S-box with non-linearity $2^{n-d-2} - 2^{n-d-1-m/2}$ but the S-box has linear structure. The second one is to construct highly nonlinear *(n, m, t, d)* resilient S-boxes without linear structure. The input for this construction are *n, m, t, d (d > m, n − d − 1 ≥ 2m).* According to the algorithm Search an *[n − d − 1, n − d − 1 − m, t + 1]* linear code $C^*$ with $C \cap C^* = \{0\}$ is applied followed by Obtain an *[n − d − 1, m, t + 1]* linear code *C*. Then obtaining $\pi$ and $T_i$ as done in Theorem 3 of the paper and Construct $h_i(y, z)$ as done in Theorem 5 of the paper are followed. Finally Output $(f1, f_2, \dots f_m)$ with $f_i(x, y, z) = x \cdot [T_i \circ \pi](y) + h_i(y, z)$ is obtained. The researchers constructed S-box with higher non-linearity of S-box with the second method than the non-linearity of S-box obtained with the first method.

All the above papers are based on Algebraic construction method. As the researchers said, with this S-box construction method we can get S-boxes having optimal cryptographic properties. But

as can be seen in the introduction part of this thesis algebraic construction method of S-box is highly vulnerable to algebraic attack [22].

## 3.2. Heuristically Designed S-boxes

On [14, 15, 17], the researchers proposed heuristics development method for developing S-box of different size. There are different approaches in heuristic methods but the main aim of the method is improving one or more cryptographic properties of S-box or Boolean function. These three papers are categorized as heuristic methods because even if their specific techniques to develop the S-box is different, their development methods are based on the techniques that the heuristic method is applied.

John and others on paper [14] are focus to develop regular *nxk* S-box which satisfies high non-linearity and low autocorrelation in order to protect linear and differential cryptanalysis respectively. First the developers generalized unusual cost function developed for single output Boolean functions for the case of S-boxes to produce significant improvements on previous work. With this method the developed 8x8 bijective S-box was not optimal, by this reason the developers change the cost function and use this new cost function with Annealing based search as a result this approach have found function that simultaneously improve non linearity and autocorrelation. With this method the developers improve the non linearity of 8x8 bijective S-box from 98 to 100.

TESAR in paper [15] proposed the method that combine genetic algorithm and total tree search to generate 8x8 S-box. In order to generate good 8x8 S-box rapidly the developers focus on parameter choosing for the algorithm and generating the new cost function: $WHS_{R,X}(P) = \sum_{b \in b^k} \sum_{w \in b^n} \left\| \hat{f}\, b(w) - X \right\|^R$ where $X$ and $R$ are real-valued parameters and $\hat{f}\, b(w)$ is the walsh hadamard transform. According to the paper, this method is divided in to 10 steps. The first step of this algorithm is to set the parameters and the non-linearity to be found the iteratively search through the tree part until the specified non-linearity is found. Finally with this method it is capable of producing 8x8 S-boxes with non-linearity 104.

In paper [17], the researcher's deals with generating S-boxes that meet the Strict Avalanche Criteria (SAC), are non-linear, and have a high degree of resistance to differential cryptanalysis. According to Anthony and others the criterion considered during the development have great impact on the resistance of two main cryptanalysis: namely Linear and differential cryptanalysis.

Initially they generated S-box randomly according to the paper, major problem with the original algorithm is that once no suitable values can be found that meet the primary criteria, a less strict set of criteria are adopted and used for the rest of the row. The developers modify the original heuristic algorithm to meet the desired criteria.

### 3.3. Evolutionary Computationally Designed S-boxes

On [16], Evolutionary Computation (EC) methods are inspired from evolutionary mechanisms such as natural selection and social and adaptive behavior. It has a common property with heuristic method is that it doesn't require objective functions with good mathematical properties. In this paper the researchers use two Evolutionary Computation methods, namely the Particle Swarm Optimization method and the Differential Evolution method to address the problem. These methods are initially applicable to real optimization problem, but according to the paper it is applied here because the performance of these methods is proved in handling discrete optimization by rounding off the real values of the solution to the nearest integer value. As result, they generated 1000 sample 8x8 S-box with non-linearity 100 and 88 autocorrelation.

### 3.4. Randomly Designed S-boxes

On [19], Grocholewska-Czuryło proposed to solve the problem associated with Boolean functions constructed with explicit constructions. The problem in explicit constructions is, functions generated with this method have particular (algebraic or combinatorial) structures, which may induce weaknesses regarding existing or future attacks. Therefore, the aim of this paper is random construction of Boolean functions with high degree of correlation immunity. A Boolean function is said to be correlation immune if and only if the function whose output leaks no information about its input value [23]. The developer reviews random bent Boolean function generator algorithms, among those, the algorithm for the generation of bent functions in ANF domain takes as its input the minimum and maximum number of ANF coefficients of every order that the resulting functions are allowed to have. This algorithm considerably reduces the possible search space. As a result the researcher uses this method by modifying this algorithm for generating highly nonlinear balanced function with higher order resiliency.

## 3.5. S-box Designed with Novel Method

On [21], the aim of this paper is to develop cryptographically strong 8x8 S-box which can avoid the use of independent key on each round. The developers use the idea of Chaotic Lorenz system to generate Chaos based algorithm for S-box design. The algorithm is divided in to two parts namely, diffusion and substitution. In the diffusion part incorporated, system trajectories are obtained by solving the Lorenz system with selected initial conditions and chaotic parameter values employing the four-step Runge–Kutta method, selected trajectory is sampled at every (number of data/256) step. In this process, the system trajectories are evaluated by the solution of the Lorenz chaotic system. Then by using the linear functional transformation, Outputs corresponding to each sample is coded starting from 0 to 255. Finally they measured the performance of the proposed S-box with five properties of S-box and compared with S-boxes generated with the methods based on the idea of Chaotic Lorenz system and AES S-box.

## 3.6. Summary

S-box development is studied in different approaches for various input output sizes. We have tried to discuss about some of the works related to our work. In section 3.1 we have seen four algebraically designed S-boxes. There are different mathematical algebraic techniques which can have a capability of generating S-box with non-linearity in the range of 104-112. Even though S-boxes of this method are optimal, Since, S-boxes developed with this method uses an algebraic equation, there is a statistical relationship between the individual components of the S-box which makes the algebraic attack effective.

Hence our aim is to develop S-box that will not have algebraic relationship among the component Boolean functions of the S-box.

We have seen three different heuristic S-box development including simulated annealing and genetic technique. These papers were aimed for the development of stronger S-box with respect to specific criterion, and then they came up with good cryptographic S-boxes. Our development method is also in heuristic S-box development category specifically dynamic hill climbing method of heuristic techniques. Instead of using the dynamic hill climbing technique to generate a single Boolean function, we tried to customize the technique in the way that optimizes specific property of the S-box.

Evolutionary computationally designed S-box [16] and randomly designed S-box [19] are discussed in section 3.3 and 3.4 respectively. Evolutionary method is somehow similar with heuristic methods on the idea that do not require objective functions with good mathematical property. The highest non-linearity achieved with this method is 100. This nonlinearity value is relatively small when it is compared with the current standard S-boxes. The random generation method aimed with design of highly non-linear Boolean function with higher order resiliency through modifying random generation algorithm of bent Boolean function.

The research in [21] developed cryptographically strong 8x8 S-box with the method of Chaotic Lorenz System to generate chaos based algorithm for S-box design. As the developers said the algorithm has both diffusion and confusion part. Finally the performance of the S-box is measured with previously generated S-box with this method and AES S-box.

# CHAPTER FOUR – The Proposed S-Box

## 4.1. Design Criteria of the S-box

S-box has a significance contribution to the security of a cipher. Specifically if we consider both components Boolean functions and S-box as a whole, then the following set of criteria should be targeted to strengthen the whole cipher.

1. Maximization of the nonlinearity [2].

   It should not be possible to express the operation of the S-box as a linear function of the inputs. This would expose the encryption algorithm to linear cryptanalysis through a process of solving a set of equations for a set of unknowns. Typically there is an inverse proportion between the size of the S-box and linearity of the S-box. As the input output size of the S-box increase, the S-box containing linearity will decrease rapidly.
2. Minimization of the largest non-trivial value in the XOR table

   This criterion is also used to resist the linear cryptanalysis over the S-box. Basically our aim to reduce the possibilities of linear cryptanalysis over our S-box is maximizing the non-linearity of the S-box but, at the end of our work, we can test the linear approximation table criterion by measuring the S-box.
3. Maximization of the algebraic order

   To check the algebraic complexity of our S-box we will also measure this criterion.
4. Minimization of the largest non-trivial correlation between linear combinations of input bits and linear combinations of output bits.

   Minimization of the largest non-trivial correlation helps to construct strong S-box against differential cryptanalysis. A small number of entries present 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.
5. No fixed points (SBOX[x] = x) or reverse fixed points (SBOX[x] =$\bar{x}$).

Figure 4.1 shows the entire architecture of the block cipher encryption with the proposed S-box. The detail explanation about each component of the cipher comes next.

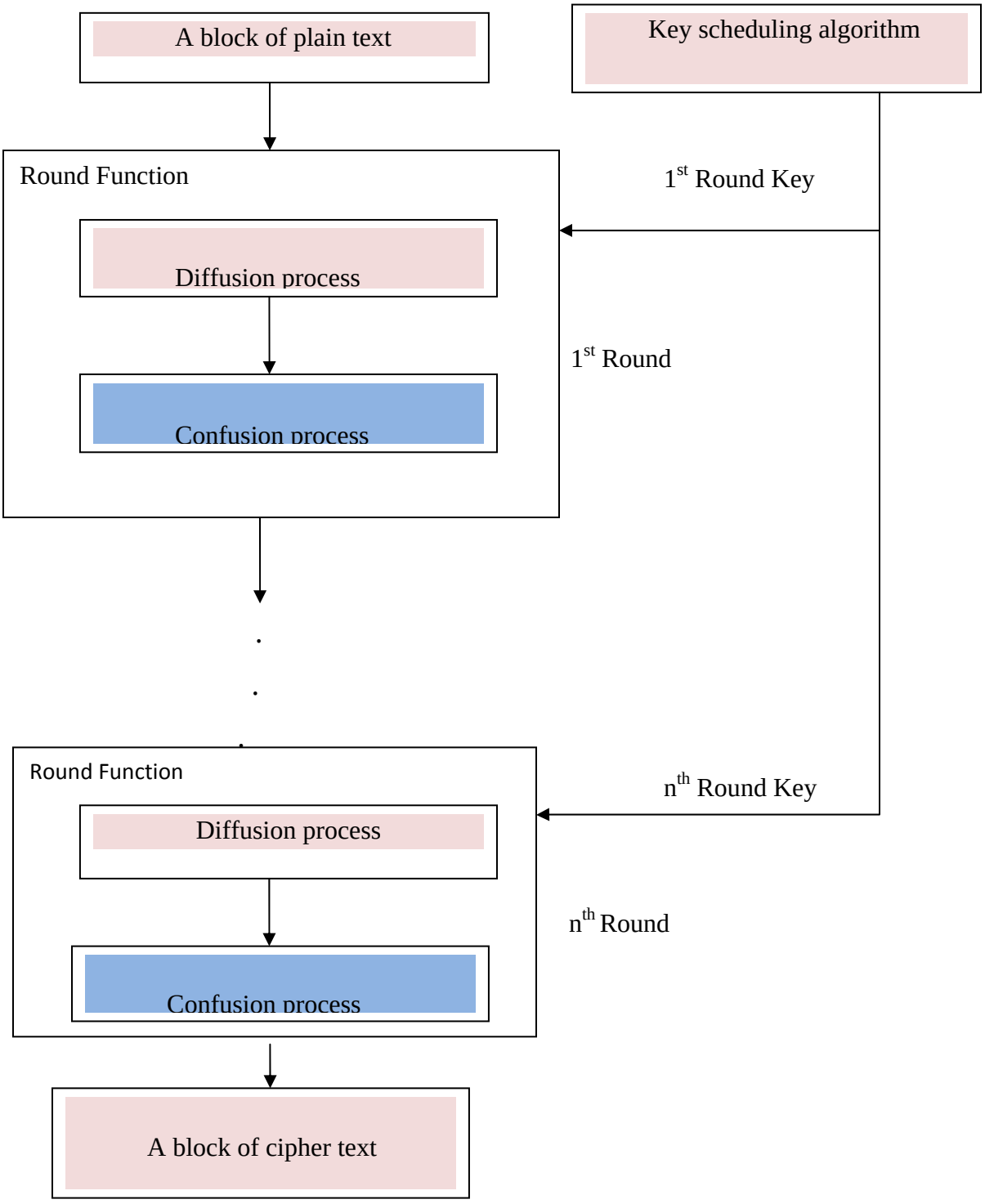


Figure 4. 1 Encryption process

## 4.2. Components of Block Encryption Algorithm

### 4.2.1. Round Function

This function contains three sub functions namely round key mixing, diffusion process and confusion process which are repeated through all rounds with different round key from the key scheduling algorithm.

### 4.2.2. Diffusion Process

In cryptography, **confusion** and **diffusion** are two properties of the operation of a secure cipher which were identified by Claude Shannon in his paper, titled with "Communication Theory of Secrecy Systems" , published in 1949.

Diffusion process is a process that makes the relationship between the plaintext and the cipher text as complex as possible. To make it complex a series of well-defined steps that can be followed as a procedure must be used as an algorithm which can manipulate the data. Mostly these steps are linear. e.g. Permutation table, Column or row rotation etc.

### 4.2.3. Confusion Process

Confusion process is a process that makes the relationship between the cipher text and the symmetric key as complex as possible. Confusion process of the encryption algorithm is the main component of the algorithm and it should be constructed to be non-linear to protect against linear cryptanalysis.

### 4.2.4. Key Scheduling Algorithm

A key scheduling algorithm is an algorithm that takes a fixed size of secret key or seed key and calculates the sub keys for all rounds. These algorithms are one way functions on which the attackers cannot derive the seed key from the round keys and also the algorithm has its own diffusion and confusion part.

Figure 4.2 shows the entire architecture to generate the proposed S-box. The detail explanation about each component in the architecture comes next.

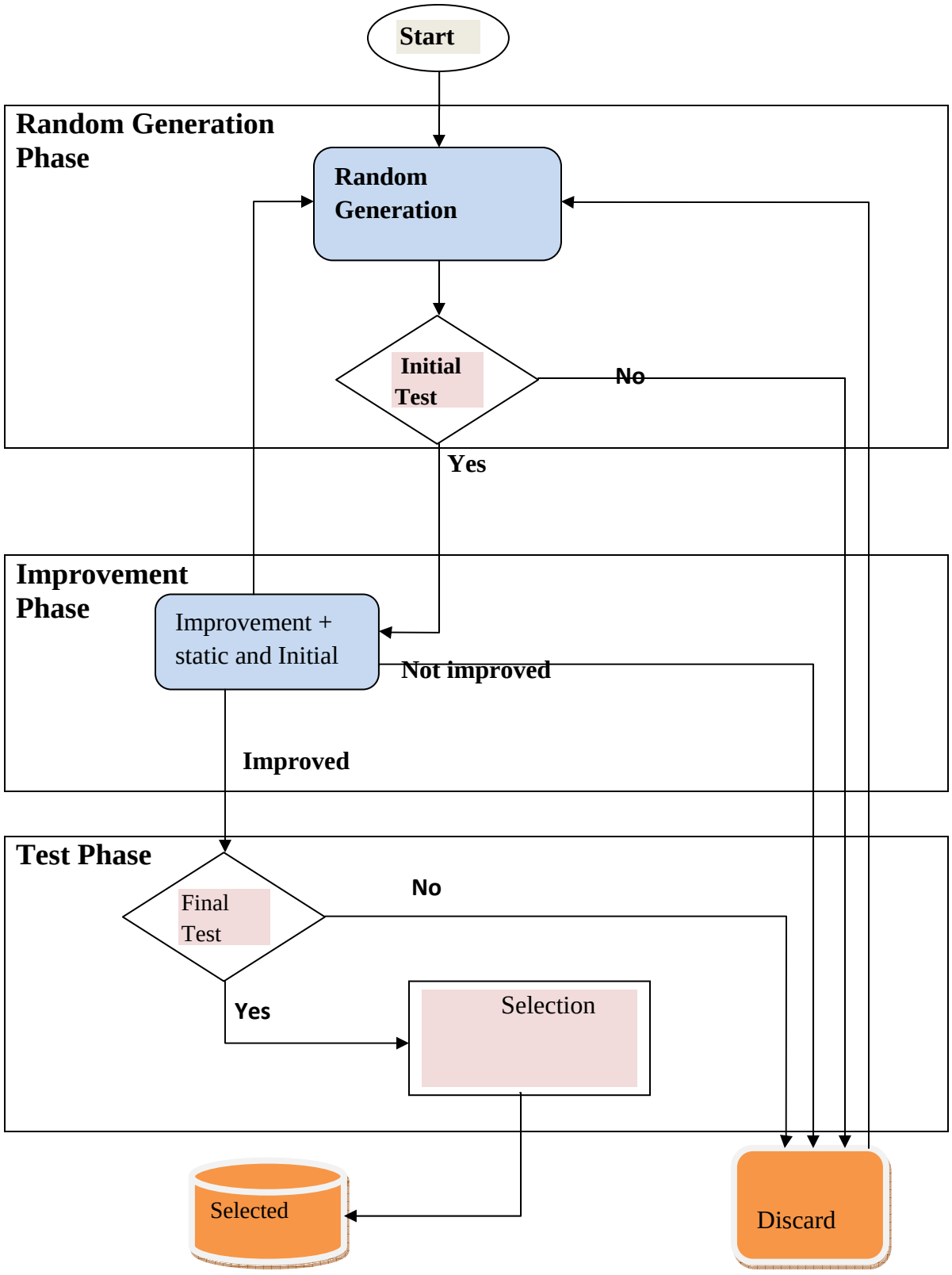


Figure 4. 2 S-box Generation Process

## 4.3. Steps of S-box Development Process

### 4.3.1. Random Generation

Random generation phase of the thesis generates a random 8x8 S-box that meets the following criteria.

The S-box should:

- Be Bijective
- Not have any fixed point
- Have minimum non-linearity of 98
- Have balanced Boolean functions

Initially we set these criteria, because the three criteria except the nonlinearity are not the type of criteria to be improved. Therefore; it is better to maintain these criteria while improving the nonlinearity criterion of the S-box. In order to minimize the computational time of improvement process, we set the minimum nonlinearity for randomly generated S-box with nonlinearity of 98. Mostly the value 98 is the maximum nonlinearity achieved by the random generation method

### 4.3.2. Initial Test

Initial test is used to check whether the random S-box has Improvement or Static set or not. As can be seen from figure 4.2; if the S-box has improvement or static set that means it will have a chance to be improved so that the S-box is entered to Improvement phase otherwise the S-box should be discarded.

### 4.3.3. Improvement

In improvement phase involves static and initial test, the dynamic hill climbing algorithm of heuristic technique for Boolean function is customized for S-box and implemented to improve the nonlinearity of the S-box. The static algorithm in this improvement set is used to increase the searching space from the local maximum to global maximum. Initial Test which is integrated with improvement is to check whether the S-box has an improvement set or static set on each iterations of the improvement process so as to terminate the iteration.

### 4.3.4. Final Test

The improved S-box from the improvement phase should be used as input for the final test. Final Test is a final test phase which includes algebraic order, LAT (linear approximation table), DDT (Data Distribution Table), Robustness, SAC (Strict Avalanche Criterion), Avalanche criterion and BIC (Bit Independence Criterion) criteria to be tested.

### 4.3.5. Selection

Selection is a process to select S-box based on measured values taken from Final Test phase and the selected S-boxes having relatively good property is selected and compared with S-boxes from literature.

### 4.3.6. Discard

This phase is a process of S-boxes which are not found at optimal level based on the measured values of Final Test and are not selected during selection process are rejected.

# CHAPTER FIVE – Implementation

## 5.1. S-box development

The objective of this work is to generate an 8x8 S-box which is stronger against known attacks and stronger with respect to known criterion of S-box. The architecture and basic explanations about the components of the system is covered in the previous chapter. Here we will look into the details of the design of our newly proposed S-box mainly discussing about the algorithms employed in the system and the rationale behind selecting the algorithms. The overall system has two major parts, the random generation part and the optimization part.

Next, we shall discuss about the implementation environment and then we will have a detail view of the sub parts in the system.

## 5.2. Development Environment

Since this environment goes to performance, for both major parts we have prepared c++ program. The random generation part takes the size of the S-box as input to generate the final S-box that maintains the given criteria of the randomly generated S-box. The optimization part can basically take the randomly generated S-box as an input, and then the program delivers the final optimized S-box when the optimization can't proceed. Therefore, random generation, optimization part and testing criteria are implemented in a system with Intel Core 2 Duo CPU of 2.39 GHz speed, a 2 GB RAM, and a Microsoft Windows XP operating system.

Next, let's go for the details of implementing every component and sub components one by one.

## 5.3. Random generation

We have customized random number generation function, rand() to generate 8 Boolean functions of size $2^8$ bits . The program generate random S-box by taking the size and number of the Boolean function as input and the four properties that we have mentioned to be the criteria of randomly generated S-box in the previous chapter as testing. Algorithm 5.1 is the customized random generation algorithm to generate random S-box that achieved the four criteria. The algorithm set each Boolean function to have 128 hamming weight to meet the balanced criteria of each Boolean functions.

***Algorithm 5.1***

***Input*** *- input output size of S-box: 8*8*

*- hwt; the Hamming weight of the desired function*

***Output*** *- S-box with the desired property*

1. *Initialize the 8 Boolean functions.*
2. *While the counter is eight:*

   *While counter2 is set 128*

   *Randomvalue = rand()%256*

   *If ( fn[counter][randomvalue] == 1)*

   *Counter2--*

   *Else fn[counter][randomvalue]=1*
3. *Generate all linear combination functions of the eight Boolean functions.*
4. *Calculate the WHT for all 256 linear combination functions and take the maximum*

   $$WHT_{max} = max(WHT)$$
5. *Calculate the nonlinearity for all linear combination functions and take the minimum of them as the nonlinearity of the S-box.*

   $$NonLinS = min\ (NonLinf\ (WHT_{max}))$$
6. *Check the fixed point of the S-box.*

   a) *Set fixedpoint to zero*

   b) *While counter is 256:*

   *If S-box[counter]==counter*

   *Set fixed point to one*
7. *If (NonLinS >=98 && fixedpoint==0):*

   *Break;*

   *Return S-box.*

Figure 5. 1Customized Random S-box Generation Algorithm

## 5.4. Optimization

This program mainly focuses on the improvement of one property of S-box; in our case nonlinearity is the property to be improved. The optimization part has two parts, improvement and testing parts. In the improvement part, we have applied two-step improvement technique as in dynamic hill climbing for the randomly generated S-box in order to maintain the balanced property of each Boolean function. Algorithm 5.2 is the optimized dynamic Hill Climbing algorithm to improve our S-box. Basically the program accepts the randomly generated S-box as input and converts it in to the Boolean functions then generates all possible linear combination of Boolean functions and takes Boolean function with minimum non linearity as input to the improvement function of the optimization. The testing part of this phase is to test whether the Boolean function has an improvement set or static set or not. If the function has no an improvement set, it has to deviate to the static set to get new Boolean function that has a chance to be improved. The testing phase has also check the non-linearity of the Boolean functions in order to select Boolean function with minimum nonlinearity. Non-linearity is implemented as stated on literature review part of this thesis.

$$N(f) = \frac{1}{2}\left(2^n - \left|\widehat{F}\,max\right|\right)$$

Non-linearity for the S-box is calculated with the same formula, but the measurement should apply for all possible linear combination of 8 Boolean functions of the S-box. Finally the non-linearity of the S-box is the minimum non-linearity of all linear combination of the Boolean functions.

***Algorithm 5.2***

***Input*** *- Randomly generated 8x8 S-box*

*- hwt; the Hamming weight of the desired function*

***Output*** *- S-box of high nonlinearity*

8. *Extract 8 Boolean functions from randomly generated 8x8 S-box and calculate non-linearity for all linear combinations of 8 Boolean functions, and take the function f(x) with minimum non-linearity of all.*
9. *Find the 2-step improvement set of f(x), such that* $f(x_1) \neq f(x_2)$ *for all pairs* $(x_1, x_2)$.
10. *While the improvement set is not empty:*
    d) *Select a random pair* $(x_1, x_2)$ *from the improvement set and complement* $f(x_1)$ *and* $f(x_2)$.
    e) *Find the new WHT of f(x).*
    f) *Find the new 2-step improvement set.*
11. *Set the variable static to zero.*
12. *Find the 2-step static set of f(x), such that* $f(x_1) \neq f(x_2)$ *for all pairs* $(x_1, x_2)$.
13. *If the static set is not empty:*
    f) *Increment static.*
    g) *Select a random pair* $(x_1, x_2)$ *from the static set and complement* $f(x_1)$ *and* $f(x_2)$.
    h) *Find the new WHT of f(x).*
    i) *Find the new 2-step improvement set.*
    j) *If the improvement set is not empty return to step 3, else return to step 5.*
14. *Return S-box.*

Figure 5. 2 Customized Dynamic Hill Climbing Algorithm

# CHAPTER SIX – Result of the Experiment

Experimental results will be provided to demonstrate the considerable impact of our optimization techniques and therefore provide a benchmark for future S-box package. For these reasons this research is a timely and useful contribution to the global effort towards the design of secure S-box.

The efficiency of the dynamic hill climbing method for generating highly nonlinear Balanced Boolean functions is clearly illustrated in [2]. The experimental result of the Dynamic Hill Climbing method compared with strong and weak hill climbing methods is also listed on Table 5.9 of [2]. But here in our case the experimental result is not for generating Boolean functions with dynamic hill climbing, our result is the effectiveness of our modified dynamic hill climbing method in generating strong bijective 8x8 S-box. For testing purpose we have implemented eleven testing criteria based on the equations in the literature review part of this thesis. As we understand from this thesis, having 8 strong Boolean functions will not lead us to get stronger S-box because the strength of the S-box is also highly dependent on the correlation of each Boolean function in addition to the strength of each individual Boolean function. We have seen the dynamic hill climbing algorithm stated in [2] can generate an individual Boolean function with nonlinearity up to 116 and S-box with nonlinearity up to 100. Therefore; since our main focus is on generating Stronger S-box, this section should contain the comparison of the newly generated S-box with those S-boxes selected for the comparison purpose.

The newly generated S-boxes are tested based on 8 main criteria's of S-box. The detail of some criteria's that we haven't discussed on literature review part will be seen as follows.

## 6.1. Linear Approximation Table

LAT is a table that shows the linearity property of the S-box. The table is constructed by calculating the parity (exclusive-or) of the subset of the input bits and the output bits for each of the possible inputs of the S-box, and counting the number of inputs whose subset's parity is zero. If the S-box is linear in the bits of the subset, all the inputs must have a zero parity of the subset.

Therefore the elements of LAT are the calculated value of the number of zero parity for each of the possible subsets of the input and the output bits of the S-box. This implies, the highest [23] absolute value in the LAT of the S-box is the most linear entry of the S-box. As much as possible, the developer wants this number to be minimized in order to protect linear cryptanalysis.

As can be seen from Table 6.1 the highest value of LAT for AES and Camellia is 16. Therefore this value is much better than that of Skipjack and Hierocrypt.

## 6.2. Differential Distribution Table

The Differential Distribution Table (DDT) [17] is a table that shows the distribution of the input EXORs and the output EXORs for all possible pairs. The table has $2^n$ number of rows which is the number of all possible input bits for n bits input S-box, and has $2^m$ number of columns (the number of all possible output bits) for m bits output S-box.

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. 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 [17]. There are two main characters of the table to be able to classify the given S-box as excellent with respect to DDT. In addition to all elements in the table are as minimum as possible, the number of zeros (impossible entries) and the number of non-zeros (possible entries) in the table should be proportional.

In order to minimize the biasedness of the S-box from differential cryptanalysis, it's good to make the largest value of DDT as minimum as possible.

According to Table 6.2, the DDT column in the DDT table refers to the highest value in the DDT table of the S-box. As a result, 4 is the value for AES's and Camellia's S-boxes and the proportion of possible and impossible entries for these S-boxes are almost equal. This makes these two Algorithms stronger against differential cryptanalysis.

## 6.3. Robustness

Robustness [17] is measured based on the number of nonzero elements, *N*, in the first column of the DDT (excluding the first element). The first column on the table denotes instances when a change in the input results in no change in the output. Such occurrences are a weakness and play an important role in differential cryptanalysis. The other feature is the largest value found in the DDT, *L*, other than the value in the first column and first row in the table (1,1). Therefore, this effect is calculated with a formula as follows:

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

Where *n* is the number of input bits. The higher *R* is, the more difficult differential cryptanalysis is to perform on the S-box.

## 6.4. Relative Error for the Avalanche Criterion and SAC

It becomes difficult to satisfy the avalanche and SAC criteria for input n=6 and larger [50]. So, it is important to express the criteria with in an error range of $[- e_A , +e_A]$, which is called "relative error interval for avalanche criterion" and "relative error interval for SAC criterion" respectively. So, relative error for the avalanche criterion is calculated as:

$$e_A = \max |2K_{AVAL}(i)-1| \; {}_{1 \le i \le n}$$

Relative error for the SAC criterion is also calculated as:

$$e_S = \max |2K_{SAC}(i,j)-1| \; {}_{1 \le i,j \le n}$$

Where $k_{AVAL}(i)$ is the probability of change of the overall output bits when only the ith bit in the input string is changed and $k_{SAC}(i,j)$ is the probability of change of the jth output bit when the ith bit in the input string is changed.

## 6.5. Experimentation

We have set nonlinearity of the S-box 102 as a condition for the termination of the improvement process because further improvement of the nonlinearity of the S-box will result on the reduction of other criteria of the S-box. We have generated 15 8x8 bijective S-boxes with no fixed point. The S-boxes are generated using five computers of the same specification. Each computer generates five S-boxes and the average optimization process time for single S-box is 86,400 seconds.

The abbreviation N-S1, N-S2… stands for the newly generated S-box1, S-box2 and so on.

Table 6.1 presents the test results and comparison of the fifteen proposed S-boxes. As a result of this comparison, *N-S1* shows better performance than that of the other.

As shown in Table 6.1 the proposed S-boxes are tested and compared with eleven testing criteria.

- **Nonlin :** is to denote the nonlinearity of the S-box. As the nonlinearity increase the strength of the S-box increase.
- **Order:** is the algebraic order of the S-box. This criterion of the S-box tells the algebraic complexity of the S-box.
- **LAT**: is the linear approximation table of the S-box. As the maximum value from the entries of the linear approximation table increases, the probability of effectiveness of linear cryptanalysis on the S-box will increase.
- **DDT**: is an abbreviation for differential distribution table of the S-box. The effectiveness probability of differential cryptanalysis will increase as the value of the DDT increase.
- **Poss. Entry and Imp. Entry:** are to denote the possible entry and the impossible entry of the S-box. An equal proportion of these values will result in cryptographically strong S-box.
- **BIC:** is the bit independence criterion of S-box.
- **MaxAC:** is used to denote the maximum autocorrelation of the S-box.

A C++ code for Nonlinearity and Avalanche testing criteria is shown in appendix C and D respectively.

Table 6. 1 comparison of proposed S-boxes with Eleven Criteria

| Name | Nonlin | Order | LAT | DDT | Poss.Entry | Imp.Entry | Robustness | SAC Error | Avalanche Error | BIC | Max AC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N-S1 | 102 | 7 | 26 | 10 | 39.93% | 60.07% | 0.9609 | 0.21875 | 0.0586 | 0.2575 | 88 |
| N-S2 | 102 | 6 | 26 | 10 | 39.80% | 60.20% | 0.9609 | 0.2500 | 0.0742 | 0.2482 | 96 |
| N-S3 | 102 | 6 | 26 | 10 | 39.78% | 60.22% | 0.9609 | 0.2813 | 0.0352 | 0.2814 | 88 |
| N-S4 | 102 | 6 | 26 | 10 | 39.69% | 60.31% | 0.9531 | 0.2500 | 0.0820 | 0.3139 | 104 |
| N-S5 | 102 | 7 | 26 | 10 | 39.64% | 60.36% | 0.9609 | 0.2188 | 0.0391 | 0.2671 | 96 |
| N-S6 | 102 | 7 | 26 | 10 | 39.77% | 60.23% | 0.9609 | 0.2500 | 0.0469 | 0.2560 | 96 |
| N-S7 | 102 | 6 | 26 | 10 | 39.55% | 60.45% | 0.9609 | 0.2188 | 0.0898 | 0.2192 | 96 |
| N-S8 | 102 | 6 | 26 | 10 | 39.74% | 60.26% | 0.9609 | 0.1875 | 0.0352 | 0.2856 | 104 |
| N-S9 | 102 | 6 | 26 | 10 | 39.63% | 60.37% | 0.9609 | 0.1875 | 0.0703 | 0.2412 | 96 |
| N-S10 | 102 | 7 | 26 | 10 | 39.69% | 60.31% | 0.9609 | 0.2500 | 0.0547 | 0.3072 | 96 |
| N-S11 | 102 | 7 | 26 | 10 | 39.59% | 60.41% | 0.9609 | 0.2500 | 0.0703 | 0.2260 | 120 |
| N-S12 | 102 | 7 | 26 | 10 | 39.75% | 60.25% | 0.9609 | 0.2188 | 0.0273 | 0.2212 | 96 |
| N-S13 | 102 | 6 | 26 | 10 | 39.79% | 60.21% | 0.9609 | 0.2188 | 0.0586 | 0.2511 | 112 |
| N-S14 | 102 | 6 | 26 | 10 | 39.88% | 60.12% | 0.9609 | 0.2500 | 0.0391 | 0.2228 | 96 |
| N-S15 | 102 | 6 | 26 | 10 | 39.63% | 60.37% | 0.9609 | 0.2188 | 0.0859 | 0.3077 | 96 |

Based on the result of the given criteria we have selected *N-S1* out of 15 newly generated S-boxes. An example of such S-box and its equivalent Boolean functions are given in appendix A and appendix B respectively.

Table 6.2 presents the comparison of our proposed S-boxes with the four S-boxes from literature with respect to eleven testing criteria. The four S-boxes selected for comparison purpose with the base of the developers. All developers are from different stream. The detail will be described as follows:

AES is developed by Joan Daernen and Vincent Rijrnen [26]. The algorithm is kept as standard algorithm for encryption and decryption by the United States government [26]. The Rijndael algorithm S-box is constructed algebraically (GF($2^8$) finite field method) and the algorithm is selected to be the first candidate by NIST as Advanced Encryption Standard among fifteen candidates.

Camellia is developed by Mitsubishi and NTT (Nippon Telegraph and Telephone). The cipher has been approved for use by the ISO (IEC) the European Union's NESSIE project and the Japanese CRYPTREC project. The S-box for this algorithm is developed algebraically with a finite field concept (GF($2^8$)) [24].

Skipjack is a block cipher developed by U.S. NSA (National Security Agency). The algorithm was originally intended for use in the controversial Clipper Chip. The development method of the algorithm as well as the S-box is kept secret by the U.S government [25].

Hierocrypt is developed by Toshiba in 2000 [27]. They were submitted to the NESSIE project, but were not selected. As they stated on [51], the design of this cipher as well as S-box is highly similar with the design of AES.

The result of the experiment shows that each of the generated S-boxes have slight difference in all criteria except the nonlinearity of the S-box but these slight difference has a big impact in cryptography. As shown in table 6.2, our proposed S-boxes are compared with four S-boxes from literatures (the first four rows). The proposed S-box *N-S1* has non-linearity 102, algebraic order 7,SAC error 0.21875, avalanche error 0.0586, BIC 0.2575, autocorrelation 88 and no fixed (or reverse fixed) points. Therefore, as shown from Table 6.2 the experimental result

demonstrates the newly generated S-boxes are much better than the S-box of Skipjack algorithm in all criteria of S-box.

Table 6. 2 Comparison of Proposed S-box with Standard S-boxes

| **Name** | **Nonlin** | **Order** | **LAT** | **DDT** | **Poss.Entry** | **Imp.Entry** | **Robustness** | **SAC Error** | **Avalanche Error** | **BIC** | **Max AC** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AES | 112 | 7 | 16 | 4 | 49.25% | 50.75% | 0.9805 | 0.1250 | 0.0352 | 0.1341 | 32 |
| Camellia | 112 | 7 | 16 | 4 | 49.47% | 50.53% | 0.9844 | 0.0938 | 0.0352 | 0.1317 | 32 |
| Skipjack | 100 | 6 | 28 | 12 | 39.94% | 60.06% | 0.9531 | 0.2188 | 0.0703 | 0.3028 | 96 |
| Hierocrypt | 106 | 6 | 22 | 6 | 47.76% | 52.24% | 0.9766 | 0.0938 | 0.0391 | 0.1860 | 72 |
| **N-S1** | **102** | **7** | **26** | **10** | **39.93%** | **60.07%** | **0.9609** | **0.21875** | **0.0586** | **0.2575** | **88** |

As shown in Table 6.2 algebraically developed S-boxes are at optimal level but according to Fuller's analysis [2] of algebraically generated S-boxes, the Boolean functions of these S-boxes are in the same class. This characteristic of the S-box leads to some statistical pattern that makes the algebraic attacks more effective on this kind of S-boxes. Whereas the components of S-box generated with heuristic methods are in different classes this makes the algebraic attack impossible on the S-box.

# CHAPTER SEVEN – Conclusion and Future Work

## 7.1. Conclusion

The goal of this thesis is to construct stronger S-box with one of heuristic technique of optimization of Boolean function, with the consequence of improving the security of the whole encryption algorithm.

We have seen representation and properties of Boolean function with their mathematical background and their application on the strength of both Boolean function and S-box. The strength of the Boolean function affects the strength of the S-box. i.e.; the strength of the S-box increases as the strength of each of Boolean function increase, but from our experiment we identified that in addition to having strong individual Boolean functions, the correlation between the Boolean function also affect the strength of S-box. Therefore the strength of their linear is used as one important factor.

We have also seen some heuristic techniques on optimization of Boolean function. The heuristic technique is a novel method in constructing and optimizing Boolean functions because, there is no algebraic relationship between Boolean functions generated with this method. Therefore, the cryptanalysts can't get any relationship among the Boolean functions.

Traditional hill climbing is important in finding Boolean functions having local maximum property. So as the S-box generated with combination of these Boolean functions will have the property of local maximum or even minimum. The second version of heuristic method is the improved version over the first version of heuristic method on its search space. Instead of searching stronger Boolean functions locally, the dynamic hill climbing algorithm tries to search stronger Boolean function all over the space. Since there is no local boundary here, the dynamic hill climbing will be much better than the traditional hill climbing in generating stronger Boolean function.

The dynamic hill climbing algorithm proposed in [2] to generate stronger Boolean function can generate strong Boolean function. But we have seen that the strength of the S-box is not only dependent on the strength of the individual Boolean functions. For example the nonlinearity

criterion of the S-box is the minimum nonlinearity of each individual Boolean functions as well as linear combination Boolean functions of the individual Boolean functions. Therefore, our customized dynamic hill climbing algorithm to generate stronger S-box, works on the Boolean function with smallest nonlinearity of all Boolean functions of the S-box including all linear combination Boolean functions.

In order to evaluate the performance of the proposed S-box, a comparison is presented by the application of strict avalanche criterion, avalanche criterion, Algebraic order, linear approximation probability, differential approximation probability, bit independent criterion, and nonlinearity analysis, maximum autocorrelation and robustness. The existing S-boxes, which are used for the purpose of benchmarking, include AES, Camellia, Skipjack, and Hierocrypt S-boxes. The results yield that the new S-box has desirable properties suitable for encryption applications for secure communications.

From this thesis we can derive several applications. These are a study of the use of computer to manipulate Boolean functions presented an opportunity to expand the boundaries of experimental feasibility, the most efficient implementation were identified and finally, identification of the testing criteria of S-box as well as implementation.

Finally this thesis fulfilled the aim to generate a stronger S-box with Boolean function concept and customize dynamic hill climbing algorithm to generate stronger S-box.

## 7.2. Future work

During this research performed, we have seen areas for future research which involve both an extension of this research, as well as topics of related work which could be investigated.

- Further optimization of the non linearity criterion of the S-box.
- Development of new heuristic method to improve the S-box's cryptographic properties.
- Construct stronger S-box with larger input bit in order the S-box to be resistance to brute force attack or exhaustive search.

# Reference


[1] An Braeken. “Cryptographic Properties of Boolean Functions and S-Boxes.” PhD thesis, Katholieke Universiteit Leuven, 2006.

[2] Fuller J.E.: “Analysis of affine equivalent Boolean functions for cryptography.” PhD thesis, Queensland University of Technology (2003).

[3] Burnett L. “Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography”, Phd Thesis, Information Security Institute, Faculty of Information Technology, QUT, 2005.

[4] Jansen C.J.A. and Boekee D.E. “The Algebraic Normal Form of Arbitrary Functions over Finite Fields.” In *Proceedings.* 8*th Symposium on Information Theory in the Benelux,* pages 69–76, May 1987.

[5] Millan W.L.. “*Analysis and Design of Boolean Functions for Cryptographic Applications*.” PhD thesis, Queensland University of Technology, Brisbane, Australia,December 1997.

[6] Preneel B., Leekwijck W. Van, Van Linden L., Govaerts R., and Vandewalle J. “Propagation Characteristics of Boolean Functions.” In *Advances in Cryptology -Eurocrypt ’90, Proceedings*, volume 473, pages 161–173. Springer-Verlag, 1991.

[7] McWilliams F.J. and Sloane N.J.A. “The Theory of Error-Correcting Codes.” North-Holland, 1978.

[8] Azizah Abd Manaf, Akram Zeki, Mazdak books.google.com.et/books?isbn=3642253261 Zamani - 2011 – Computers.

[9] Willi Meier and Othmar Sta_elbach. “Nonlinearity criteria for cryptographic functions.” In Advances in Cryptology - EUROCRYPT '89, volume 434 of Lecture Notes in Computer Science, pages 549{562. Springer Verlag, 1990.

[10] Siegenthaler T., “Correlation-immunity of nonlinear combining functions for cryptographic applications”, *IEEE Trans. Inform. Theory*, vol. IT-30, no. 5, pp. 776–780,1984.

[11] Jennifer Miuling Cheung, “*THE DESIGN OF S-BOXES*” San Diego State UniversityJM Cheung - 2010 - sdsu-dspace.calstate.edu.

[12] Mar P. P., Latt K. M., "New analysis methods on strict avalanche criterion of S-boxes", World Academy of Science, Engineering and Technology 48, 2008, . pp.150-154.

[13] Hussain, I., Shah, T., M.A, Gondal, W.A. Khan, 2011. "Construction of Cryptographically Strong 8x8 S-boxes", World Applied Science Journal, 13 (11): 2389-2395.

[14] Clark, J.A. et al., 2005. "The Design of S-Boxes by Simulated Annealing Evolutionary Comput"., 23(3):219-231.

[15] TESAŘ P . "A New Method for Generating High Non-linearity S-Boxes. "RADIOENGINEERING, 2010 - radioeng.cz.

[16] Laskari E.C., Meletiou G.C., Vrahatis M.N., "Utilizing evolutionary computation methods for the design of s-boxes", in Proceedings of International Conference on Computational Intelligence and Security, pp.: 1299–1302, 2006.

[17] Anthony Lineham and T. Aaron Gulliver. "Heuristic S-box Design", Contemporary Engineering Sciences, Vol. 1, 2008, no. 4, 147 – 168.

[18] Abuelyaman Eltayeb and El-Affendi Mohamed "A Real Time S-Box Construction Using Arithmetic Modulo Prime Numbers" Journal of Digital Information Management, Vol 5, No. 6, pp 354-360, December 2007.

[19] Anna Grocholewska-Czuryło. "Random generation of Boolean functions with high degree of correlation immunity."

[20] Shaojing Fu · Kanta Matsuura · Chao Li · Longjiang Qu. "Construction of highly nonlinear resilient S-boxes with given degree" S Fu, K Matsuura, C Li, L Qu -Designs, Codes and Cryptography, 2012 – Springer.

[21] Majid Khan · Tariq Shah · Hasan Mahmood ·Muhammad Asif Gondal · Iqtadar Hussain . "A novel technique for the construction of strong S-boxes based on chaotic Lorenz systems. "Nonlinear Dyn (2012) 70:2303–2311 DOI 10.1007/s11071-012-0621-x.

[22] Iqtadar Hussain • Tariq Shah • Hasan Mahmood • Muhammad Asif Gondal . "A projective general linear group based algorithm for the construction of substitution box for block ciphers" Neural Comput & Applic(2012) DOI 10.1007/s00521-012-0870-0.

[23] Aoki K, lehikawa T, Kanda Metal. "Specification of Camellia – A128 bit block cipher." In proc. Selected Areas in cryptography (SAC'2000), Waterloo, Canada, LNCS 2012.

[24] Schneier B., "Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)," *Fast Software Encryption: Second International Workshop, Leuven, Belgium, December 1994, Proceedings*, Springer-Verlag, 1994, pp. 191–204.

[25] Daemen, J., Rijmen, V.: The design of Rijndael: "AES – The Advanced Encryption Standard." Springer, Heidelberg (2002) ISBN 3-540-42580-2.

[26] Barreto P., Rijmen V., Nakahara J. Jr., Preneel B., Vandewalle J., and Kim H.Y. "Improved SQUARE attacks against reduced-round HIEROCRYPT". *Fast Software Encryption 2001*, Springer-Verlag, to appear.

[27] 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.

[28] Dalai DK, Gupta KC, Maitra S (2004). "Results on Algebraic immunity for cryptographically significant boolean functions." In: *INDOCRYPT 2004*, pp 92–106, number 3348, Lecture Notes in Computer Science, Springer-Verlag.

[29] "Construction of DES_like S_boxes Based on Boolean Functions Satisfying the SAC".

[30] Biham E. and Shamir A. "differential cryptanalysis of DES-like cryptosystems." In advances in cryptology – crypto '90, proceedings, volume 537 of lecture notes in computer science, pages 2-21. Springer – verlag, 1991.

[31] Ibrahim, S., Maarof, M.A., Ngadiman, M.S.: "Practical Security against Differential Cryptanalysis for Extended Feistel Network." Universiti Teknologi Malaysia.

[32] Feistel H. "Cryptography and computer privacy." Scienti_c American, 228(5):15{23, 1973.

[33] Forré R., "The strict avalanche criterion: Spectral properties of Boolean functions and an extended definition", Proceedings of Crypto'88.

[34] Kim K., Matsumoto T., Imai H., "A recursive construction method of S-boxes satisfying strict avalanche criterion," *Advances in Cryptology, Proc. Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 564–575.

[35] Carlet C.: "Boolean functions for cryptography and error correcting codes." In: Crama Y., Hammer P. (eds.) Boolean Methods and Models. Cambridge University Press, Cambridge (in press).

[36] Dobbertin H. “Construction of bent functions and balanced Boolean functions with high nonlinearity.” In *Fast Software Encryption*, number 1008 in Lecture Notes in Computer Science, pages 61–74. Springer-Verlag, 1994.

[37] Regazzoni F., Eisenbarth T., Grosschlä J. dl, Breveglieri L., Ienne PP., Koren I., Paar C., “Power Attacks Resistance of Cryptographic S- boxes with added Error Detection Circuits”, Defect and Fault Tolerance in VLSI Systems (DFT 2007), pp 508-516, 2007 .

[38] Mister, S., Adams, C.: “Practical S-Box Design.” In: Workshop on Selected Areas in Cryptography (SAC 1996) Workshop Record, pp. 61–76. Queens University (1996).

[39] Millan, W., “How to Improve the Non-linearity of Bijective S-boxes”, in*3rd Australian Conference on Information Security and Privacy* (Boyd, C. and Dawson, E. eds.),*LNCS, 1438*, pp. 181–192, Springer-Verlag, April 1998.

[40] Kam, J.B., and Davida, G.I.: “Structured Design of Substitution Permutation Encryption Networks.” IEEE Transactions on Computers, Vol 28, No. 10, 747 (1979).

[41] Adams C. M. and Tavares S. E.. “The structured design of cryptographically good S-boxes.”*Journal of Cryptology*, 3(1):27–41, 1990.

[42] Canteaut, A., Daum, M., Leander, G., and Dobbertin, H. “Normal and nonnormal bent functions. “ In Proceedings of the 2003 International Workshop on Coding and Cryptography (WCC 2003) (2003), pp. 91–100.

[43] Vergili I., Yücel M. D., “Avalanche and Bit Independence Properties for the Ensembles of Randomly Chosen nxn S-Boxes”, Turk J Elec Engin, Vol.9, No.2, TÜBİTAK, 2001, pp. 137-145.

[44] William Millan, Andrew Clark, and Ed Dawson. “Boolean function design using hill climbing methods.” In Information Security and Privacy, ACISP '99, volume 1587 of Lecture Notes in Computer Science, pages 1{ 11. Springer Verlag, 1999.

[45] http://securitycerts.org/review/cryptography-confidentiality.htm.

[46] Wegener I. (1987): “The Complexity of Boolean Functions”, John Wiley and Sons.

[47] http://users-cs.au.dk/bromille/dKS08/lec2.pdf.

[48] Stallings, W.: “Cryptography and Network Security: Principles and Practices”, 3rd edn. Prentice Hall, Englewood Cliffs (2003).

[49] Vergili I., Yücel M. D., “Avalanche and Bit Independence Properties for the Ensembles of Randomly Chosen nxn S-Boxes”, Turk J Elec Engin, Vol.9, No.2, TÜBİTAK, 2001, pp. 137-145.

[50] Ohkuma K., Muratani H., Sano F., and Kawamura S., “The Block Cipher Hierocrypt”,*Workshop on Selected Areas in Cryptography-SAC 2000*, Lecture Notes in Computer Science 2012, Springer-Verlag, pp. 72–88, 2001. 164, 171, 174.

[51] Webster A.F., Tavares S.E., “On the design of S-boxes”, Proceedings of Crypto’85, pp. 523–534.

# Appendices

## Appendix A: Sample S-box generated with Optimized Dynamic Hill Climbing

The following S-box is S1 from the Table 6.1. It has nonlinearity 102, algebraic order 7, contains no fixed points. The S-box has DDT of 10.

**S1[256]=**

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

## Appendix B: The equivalent Boolean functions for the S-box in appendix A

**f1[256]**=1110010111001101000010001110011001000110110001010011101111010001001
0001011101101001001010000111101101001011111000000100101110010111001100101011
1011100001100001100110001101001010001111001101111001110001011011100001100000
0000001100011111110111110010010111101 01

**f2[256]**=0111110010110111110101101001010000100010100100001101010010000001011
0001101100111110110010111101000101111000100010110111011111010000101000111111
1101110110011001101100000000110010110100001011101110110111001010001100101010
0100000101010101010101010011100011100

**f3[256]**=0011110011101010111000101011111010001101001011110010000100100011000
0011100000110100000110100000100101011111101101100100010100100000101010110100
0010110100111111100101010100001111111011101001100111010101111101111101111110
0000100000001111110000101000001011010 11

**f4[256]**=0101010111010111011101110111010000111110000110000110011001011011111
1100001001100100011001000010011010011001001110100100001100001001000110110111
1110100110110100101001101001100101010101111000001001101111110100110000111001
0011001100100101100111001011101000100 1

**f5[256]**=0101110110001100100111011101011100001000111110110011111011100101111
0011011000011110100001101110110101000111110101110001001001010000000000100111
1111011010111100100111100011000100101010010100110011011011100010000100001000010
1110100010100100100001100110100100110 10

**f6[256]**=0000100010010001110110101110001000000101100110001001111100001011111
1111101111100110100110101000011110001110110100000010111010110110110001000110
0011011101110100101110001101110011001010111100000000101110001010110011001100
0010110001110001110010111100011010000 11

**f7[256]**=0000101000010100000110000011011011001000010000100101111110111111101
0110001101010111000111000110110101110000011110010111111011001001111100111100
0100011110000111100101101010101000100111001000111001100100010101011101011010
0011000001010001111101101101000010101011

**f8[256]**=1111000011011100000111100010000100101000001110100010000001001100000
1000011101000001001000111110010001001100110110010110000001111111101011111111
0011010010001010110110110101011110111011110000011101010111000101111001101010
0011010110111001011010111100011000111 10

## Appendix C: Nonlinearity Test

```
using namespace std;

bitset<256> temp;
```

```
bitset<8> temp1;
genx1(x1);
genx2(x2);
genx3(x3);
genx4(x4);
genx5(x5);
genx6(x6);
genx7(x7);
genx8(x8);
for(int m=0;m<256;m++)
{temp1=bitset<8>(sbox[m]);
f8[m]=temp1.test(0);
f7[m]=temp1.test(1);
f6[m]=temp1.test(2);
f5[m]=temp1.test(3);
f4[m]=temp1.test(4);
f3[m]=temp1.test(5);
f2[m]=temp1.test(6);
f1[m]=temp1.test(7); }
for (int k=1;k<256;k++)
{for (int i=0;i<256;i++)
{

t[i]=(x1[k]&&f1[i])^(x2[k]&&f2[i])^(x3[k]&&f3[i])^(x4[k]&&f4[i])^(x5[k]&&f5[i])^(
6[k]&&f6[i])^(x7[k]&&f7[i])^(x8[k]&&f8[i]);
                temp.set(i,t[i]);
        }
        for (int a=0;a<256;a++)
```

```
                fn[a]=temp.test(a);
            WHT(wht);
            WHTmax=max(wht);
            c=nonLin(WHTmax);
            MM[k]=c;
      if ((k==1)|| (k==2)||(k==4)||(k==8)|| (k==16) ||(k==32)|| (k==64)|| (k==128) )

            {  cout<<"The nonlinearity of   f"<<v<<" is "<<c<<endl;
               v++;}
        }
        int min=abs(MM[1]);
        for (int i=1;i<255;i++)
        {
            if (min >abs(MM[i]))
                min=abs(MM[i]);
        }
        cout<<"The min Nonlinearity of the sbox is "<<min<<endl;
        }
int nonLin(int x)
{
        return .5*(256-x);
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ max function
int max(int tp[] )
{
        int max=abs(tp[0]);
        for (int i=0;i<256;i++){
```

```
            if (max < abs(tp[i]))
                max=abs(tp[i]);
        }
        return max;
}
void WHT(int temp[])
{
        for (int i = 0;i < 256;i++) // for calcualting the truth table of linear function and the WHT of the given function
        {
            int linFn=0, pttLinFn=0, pttFn=0, sum = 0, pro = 1;
            for (int j = 0;j < 256;j++)
            {
                linFn=(x1[i] && x1[j]) ^ (x2[i] && x2[j])^(x3[i] && x3[j]) ^ (x4[i] && x4[j])^(x5[i] && x5[j]) ^ (x6[i] && x6[j])^ (x7[i] && x7[j]) ^ (x8[i] && x8[j]);
            pttLinFn = 1 - (2 * linFn);// calculating polar truth table  of the linear function
            pttFn = 1 - (2 * fn[j]); // calculating the polar truth table of the the given function
                pro = pttLinFn * pttFn;
                sum = sum + pro;// calculating the WHT of the function
            }
            temp[i] = sum;
        }
}
void genx1(int A[])
{
        int i,j,k;
        for ( i = 0;i < 256;i++) {
            A[i]=1;
```

```
        }
        for (j=0;j<256;j=j+256){
                for (k=0;k<128;k++)
                        A[j+k]=0;
        }
}
void genx2(int A[] )
{
        int i,j,k;
        for ( i = 0;i < 256;i++) {
                A[i]=1;
        }
        for (j=0;j<256;j=j+128){
                for (k=0;k<64;k++)
                        A[j+k]=0;
        }
}
void genx3(int A[])
{
        int i,j,k;
        for ( i = 0;i < 256;i++) {
                A[i]=1;
        }
        for (j=0;j<256;j=j+64){
                for (k=0;k<32;k++)
                        A[j+k]=0;
        }
```

```
}
void genx4(int A[])
{
	int i,j,k;
	for ( i = 0;i < 256;i++) {
		A[i]=1;
	}
	for (j=0;j<256;j=j+32){
		for (k=0;k<16;k++)
			A[j+k]=0;
	}
}
void genx5(int A[])
{
	int i,j,k;
	for ( i = 0;i < 256;i++) {
		A[i]=1;
	}
	for (j=0;j<256;j=j+16){
		for (k=0;k<8;k++)
			A[j+k]=0;
	}
}
void genx6(int A[])
{
	int i,j,k;
	for ( i = 0;i < 256;i++) {
```

```
                A[i]=1;
        }
        for (j=0;j<256;j=j+8){
                for (k=0;k<4;k++)
                        A[j+k]=0;
        }
}
void genx7(int A[])
{
        int i,j;
        for (j=0;j<256;j++)
                A[j]=1;
        for ( i = 0;i < 256;i=i+4) {
                A[i]=0;
                A[i+1]=0;
        }
}
void genx8(int A[])
{
        int i,j;
        for ( i = 0;i < 256;i=i+2) {
                A[i] = 0;
        }
        for (j=1;j<256;j=j+2){
                A[j] = 1;
        }
}
```

**Appendix D: Avalanche test of the S-box**

```
using namespace std;
bitset<8> num;
int temp=0;
for (int k=1;k<=128;k=2*k)
{
float kh1=0.0,kh2=0.0;
        hh1=0,hh2=0,hh3=0,hh4=0,hh5=0,hh6=0,hh7=0,hh8=0;
        for (int i=0;i<256;i++)
                x1[i]=i;
        for (int i=0;i<256;i++)
                x2[i]=(i^k);
        for (int i=0;i<256;i++)
        {
                temp =sbox[x1[i]]^ sbox[x2[i]];
                num=bitset<8>(temp);
                f8[i]=num.test(0);
                f7[i]=num.test(1);
                f6[i]=num.test(2);
                f5[i]=num.test(3);
                f4[i]=num.test(4);
                f3[i]=num.test(5);
                f2[i]=num.test(6);
                f1[i]=num.test(7);
        }
        for (int m=0;m<256;m++)
        {
                if (f1[m]==1)
                        hh1++;
                if (f2[m]==1)
                        hh2++;
                if (f3[m]==1)
                        hh3++;
                if (f4[m]==1)
                        hh4++;
                if (f5[m]==1)
                        hh5++;
                if (f6[m]==1)
                        hh6++;
                if (f7[m]==1)
                        hh7++;
                if (f8[m]==1)
                        hh8++;
        }
```

```
                kh1=hh1+hh2+hh3+hh4+hh5+hh6+hh7+hh8;;
                kh2=kh1/2048;
                kh2=2*kh2-1;
                if(max1<fabs(kh2))
                max1=fabs(kh2);



        }
cout<<"Avalanche error of the s-box is (max) "<<max1;

}
```

# Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

**Declared by:**

Name: Haymanot Adane Yimam

Signature: _____

Date: _____

**Confirmed by advisor:**

Name: Dr. Dejene Ejigu

Signature: _____

Date: _____