Analysis and Design of Some Cryptographic Boolean Functions

Ziad Saber

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical Engineering) at

Concordia University

Montreal, Quebec, Canada

December 2005

Your file Votre référence
ISBN: 0-494-14279-0
Our file Notre référence
ISBN: 0-494-14279-0

Abstract

Analysis and Design of Some Cryptographic Boolean Functions

Ziad Saber

Boolean functions are vital components of symmetric-key ciphers such as block ciphers, stream ciphers and hash functions. When used in cipher systems, Boolean functions should satisfy several cryptographic properties such as balance, high nonlinearity, resiliency and high algebraic degree.

Bent functions achieve the maximum possible nonlinearity and hence they provide optimal resistance to several cryptographic attacks such as linear and differential cryptanalysis.

We present some simple constructions for binary bent functions of length 2^{2k} using a known bent function of length 2^{2k-2}.

Adams and Tavares introduced two classes of bent functions: bent-based bent functions and linear-based bent functions. In this thesis we explore different bent-based constructions. In particular, we show that all nonlinear resilient functions with maximum order resiliency are either bent-based or linear-based. We provide an explicit count for the number of such resilient functions that belong to both classes. We also provide a simple proof that all symmetric functions that achieve the maximum possible nonlinearity are bent-based. In particular, for n even, we have 4 bent-based bent functions. For n odd, we also have 4 bent-based functions. We also prove that there is no bent-based homogeneous functions with algebraic degree >2.

Almost all cryptographic properties of Boolean functions can be determined efficiently from its Walsh transform. In this thesis, we present some restrictions on the partial sum of the Walsh transform of binary functions.

In several parts of the thesis, we extend the obtained results to functions defined over GF(p).

Acknowledgments

My heartfelt thanks and earnest gratitude are due to Dr. Amr Youssef and Dr. Walaa Hamouda for their constant support and guidance and for their enduring patience.
I'd like also to express my appreciation to all the faculty and people at Concordia University who contributed to my success one way or the other.

My deepest love and supreme appreciation and gratitude to my parents, Arafa and Imtiaz, for their unceasing love and relentless patience and devotion raising me, my brother and sisters, and for sacrificing their comfort and happiness for the sake of ours.
My thanks go to my brother Mohammad and my sisters Suzan, Andaleb, Soha, Iman as well, for their love, encouragement and support.

I am particularly indebted to Bassem Al-Taib, Eng. Khaled Jarkas, Eng. Khaled Hijazi, Roland Charaf, Eng. Haitham Hijazi, Ali Charaf, Eng. Bassam Derbass, Baraa Derbass, M. Alhazouri, Eng. R Zein, Eng. Akram Friegh, Bilal Oglo, Baher Falah, Walid Sabri, Nuha Alyassir, and Dina Khaled, whom I can't thank enough for their continual encouragement and unvarying support and assistance, and for their true friendship and loyalty through the thick and thin.

To my wife Abir that she was so patient and for giving me her love, tender and caring.
Finally, to all the people who impacted my life positively: *Thank you.*

Table of Contents

List of Figures

List of Tables

List of Notations

Z_2	The set of binary numbers
Z_2^n	The set of binary *n* tuples
$\oplus$	XOR operation
x	A variable in Z_2^n or GF(2)
$wt(f)$	The Hamming weight of f
$d(f,g)$	The Hamming distance between two Boolean function *f* and *g*
CI	Correlation immunity
m	Order of resiliency
$ANFD$	Algebraic normal form degree
NL	The nonlinearity of f
$\omega \cdot x$	The dot product of ω and x over GF(2)
$F(\omega)$	Walsh Hadamard Transform
BB	Bent-based function
LB	Linear-based function
AI	Algebraic immunity

Chapter 1

Introduction to Cryptography

1.1 Introduction

The terms *cryptology* and *cryptography* have long been used by researchers interchangeably. Only in the last half century has the modern definition of cryptology become widely adopted. This modern definition classifies cryptology as a general term that encompasses cryptography and cryptanalysis as its main component classes.

Historically, cryptography was all about encryption. The purpose of cryptanalysis was to decipher encrypted messages. Development in either of these two fields reflects development in the other. As methods of cryptography improve, the need for better methods of cryptanalysis grows. Conversely, as cryptanalysts become more skillful in breaking messages, cryptographers feel the need for better ways to encipher them.

The word for cryptography in Greek consists of two parts which mean "hidden" and "writing"; that is the art and science of secret writing. Cryptography is the practice of enciphering a message so that it can be read by someone for whom it is intended, but not by anyone for whom it is not intended. More precisely, cryptography is the process of providing secure communications over insecure channels. This definition emphasizes the fact that people often have to transmit secret messages by means of communications systems like e-mails, telephone and telegraph lines to which many people have access.

1.2 A Brief History

Cryptography is one of the oldest fields of technical study we can find records of, going back at least 4,000 years [1]. Some Egyptians used hieroglyphics to decorate the tombs of deceased rulers and kings. These hieroglyphics told the story of the life of the king and proclaimed the great acts of his life. They were purposefully cryptic, but not apparently intended to hide the text. Rather, they seem to have been intended to make the text seem more regal and important.

The ancient Chinese used the ideographic nature of their language to hide the meaning of words. Messages were often transformed into ideographs for privacy, but no substantial use in early Chinese military conquests is apparent. Genghis Khan, for example, seems to never have used cryptography.

In India, secret writing was apparently more advanced, and the government used secret codes to communicate with a network of spies spread throughout the country. Early Indian ciphers consisted mostly of simple alphabetic substitutions often based on phonetics. Some of these were spoken or used as sign language. This is somewhat similar to "pig Latin" (igpay atinlay) where the first consonant is placed at the end of the word and followed by the sound "ay".

Julius Caesar used a system of cryptography (i.e. the 'Caesar Cipher') which shifted each letter 2 places further through the alphabet (e.g. *Y* shifts to *A*, *R* shifts to *T*, etc.). The distance of the displacement is not important to the scheme, and in fact, neither is the lexical ordering chosen. The general case of this sort of cipher is the "monoalphabetic substitution cipher" wherein each letter is mapped into another letter in a one to one fashion.

The Arabs were the first to make significant advances in cryptanalysis. An Arabic author, Qalqashandi, wrote down a technique for solving ciphers which is still used today. The technique is to write down all the ciphertext letters and count the frequency of each symbol. Using the average frequency of each letter of the language, the plaintext can be written out. This technique is powerful enough to cryptanalyze any monoalphabetic substitution cipher if enough cyphertext is provided

In 1948, Shannon published "A Communications Theory of Secrecy Systems" [2] . Shannon was one of the first modern cryptographers to attribute advanced mathematical techniques to the science of ciphers. Although the use of frequency analysis for solving substitution ciphers began many years earlier, Shannon's analysis demonstrates several important features of the statistical nature of language that makes the solution to nearly all previous ciphers very straightforward. Perhaps the most important result of Shannon's famous paper is the development of a measure of cryptographic strength called the 'unicity distance'. The unicity distance is a number that indicates the quantity of ciphertext required in order to uniquely determine the plaintext of a message. It is a function of the length of the key used to encipher the message and the statistical nature of the plaintext language.

1.2 Cryptographic Goals

Of all the information security objectives, the following four form a framework upon which the others will be derived: (1) privacy or confidentiality (2) data integrity (3) authentication and (4) non-repudiation [3] .

1. Confidentiality is a service used to keep the content of information from all but those authorized to have it. Secrecy is a term synonymous with confidentiality and privacy. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible.

2. Data integrity is a service which addresses the unauthorized alteration of data. To assure data integrity, one should have the ability to detect data manipulation by unauthorized parties. Data manipulation includes insertion, deletion, and substitution.

3. Authentication is a service related to identification. This function applies to both entities and information itself. Two parties entering into communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. For these reasons this aspect of cryptography is usually subdivided into two major classes: entity authentication and data origin authentication. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed).

4. Non-repudiation is a service which prevents an entity from denying previous

commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary. For example, one entity may authorize the purchase of property by another entity and later deny such authorization was granted. A procedure involving a trusted third party is needed to resolve the dispute.

A fundamental goal of cryptography is to adequately address these four areas in both theory and practice.

1.3 Basic Encryption/Decryption Systems

Let M denote a set called the message space. An element of M is called a plaintext message or simply a plaintext.

Let C denote a set called the ciphertext space. C consists of strings of symbols from an alphabet of definition, which may differ from the alphabet of definition for M. An element of C is called a ciphertext.

K denotes a set called the key space. An element of K is called a key. Each element $e \in K$ uniquely determines a bijection from M to C, denoted by E_e. E_e is called an encryption function or an encryption transformation. For each $d \in K$, D_d denotes a bijection from C to M. D_d is called a decryption function or decryption transformation.

An encryption scheme consists of a set $\{E_e : e \in K\}$ of encryption transformations and a corresponding set $\{D_e : d \in K\}$ of decryption transformations with the property that for each $\{e \in K\}$ there is a unique key $\{d \in K\}$ such that $D_d = E_e^{-1}$. The keys, e and d, in the preceding definition are referred to as a key pair and sometimes denoted by (*e*, *d*). Note that *e* and *d* need not be the same as in public key cryptography [3].

Basic blocks of an encryption/decryption system are shown in Figure1.1. The plaintext is encrypted and transmitted over insecure channel. The cryptanalyst is an entity in this two-party communication which is neither the sender nor receiver, and which tries to defeat the information security service being provided between the sender and the receiver. The cryptanalyst can listen to, delete, insert, or modify the transmitted messages.

One can think of the key as a compact way to specify the encryption transformation (from the set of all encryption transformations) to be used. For example, a transposition cipher of block length t has $t! = t \times (t-1) \times (t-2).....2 \times 1$ encryption functions to select from. Each can be simply described by a permutation which is called the key. The size of the key space is the number of encryption/decryption key pairs that are available in the cipher system. It is a great temptation to relate the security of the encryption scheme to the size of the key space. The following statement is important to remember. A necessary, but usually not sufficient, condition for an encryption scheme to be secure is that the key space be large enough to preclude exhaustive search.

1.4 Cryptographic Techniques

Cryptographic techniques are typically divided into two types: symmetric-key and public-key (see Figure 1.2).

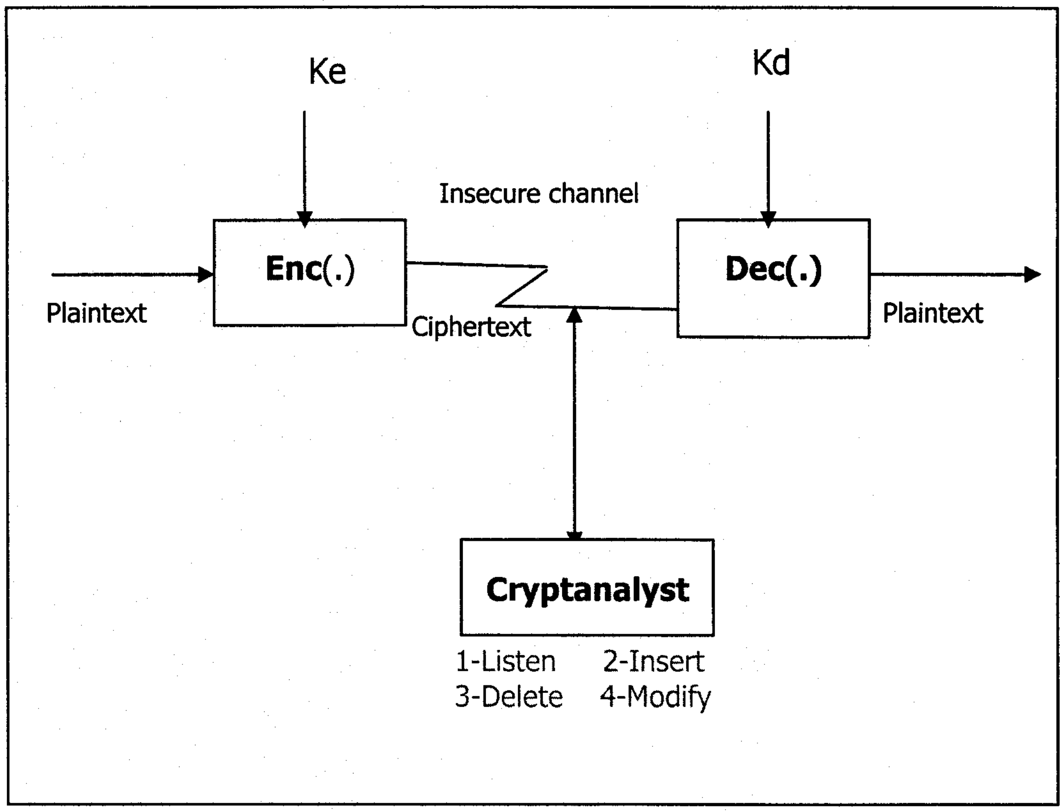

Figure 1.1: A typical encryption/decryption system.

1.4.1 Symmetric-Key Encryption

A cryptographic system is said to be a symmetric-key system if the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message. The main drawback is that the two parties should somehow exchange the key in a secure way.

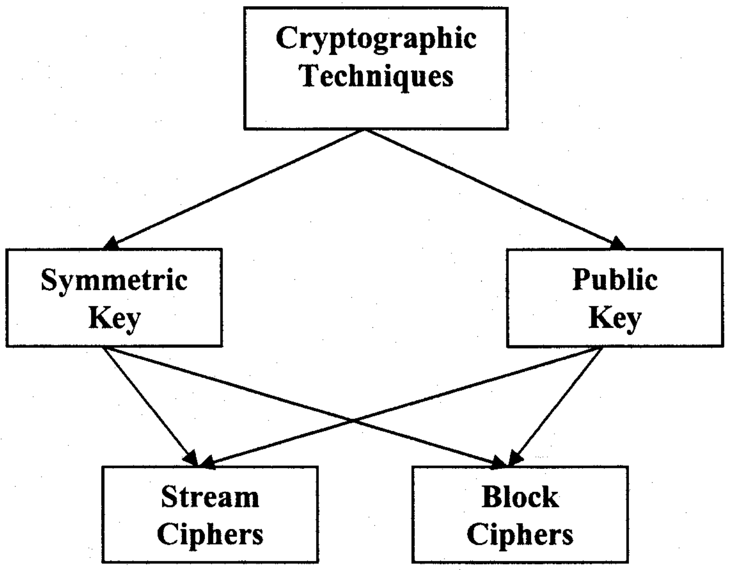

Figure 1.2: General classification of Encryption systems.

1.4.2 Public-Key Encryption

A cryptographic system is said to be a public-key system if it uses two keys, a public key and a private key. A public key is known to everyone and a private or secret key is known only to the recipient of the message. An important element to the public key system is that the public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt them. Moreover, it is very difficult to deduce the private key if you know the public key.

Public key cryptography [3] was invented in 1976 by Whitfield Diffie and Martin Hellman. It is also called asymmetric encryption because it uses two keys instead of one key (symmetric encryption).

Each of the symmetric key and the public key are classified into two classes: stream ciphers and block ciphers.

1.4.3 Stream Ciphers

A stream cipher produces a pseudo-random sequence of bits which are exclusive-OR'ed with the plaintext to produce the ciphertext. Many stream ciphers make use of the linear feedback shift register (LFSR). Figure 1.3 illustrates a linear feedback shift register defined by the primitive polynomial
$f(x) = 1 + x^5 + x^6$.

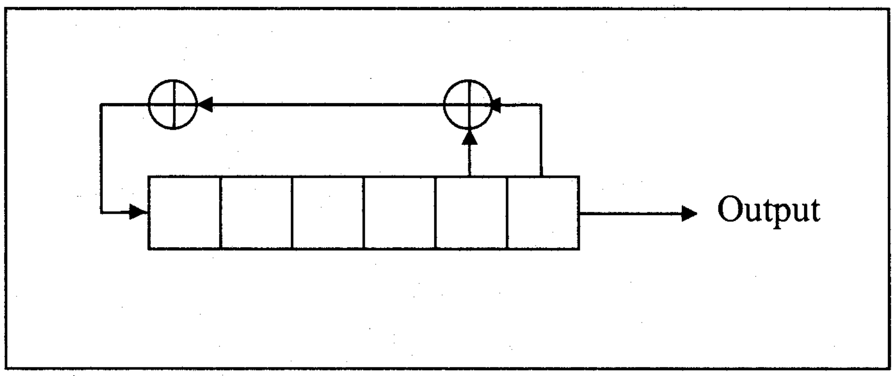

Figure 1.3: An example for a 6-bit linear feedback shift register.

Recall that a periodic LFSR is defined by a feedback polynomial of degree *L*, the length of the LFSR. When the feedback polynomial is primitive and of degree *L*, the output sequence of a maximum length LFSR is periodic with period $2^L - 1$ and is called an *m*-

sequence. One particular example of LFSR-based stream ciphers is the nonlinear combiner (see Figure 1.4) which combines the output of *k* LFSR's using a nonlinear Boolean function to obtain the keystream. Thus, the combiner requires a nonlinear Boolean function of *k* inputs. The keying material for this cipher is generally the initial contents of the LFSR's. In some cases the feedback polynomials are assumed to be public knowledge, along with the combining function. A cryptographic weakness with some LFSR-based stream ciphers is that the output sequence from the LFSR is correlated to the output keystream sequence of the generator. Hence, the design of these non-linear combining Boolean function is of great importance in stream cipher in order to stand against different cryptanalytic attacks. In other words, if the nonlinear combining function is not properly designed, then an attacker may be able reconstruct the keystream sequence.

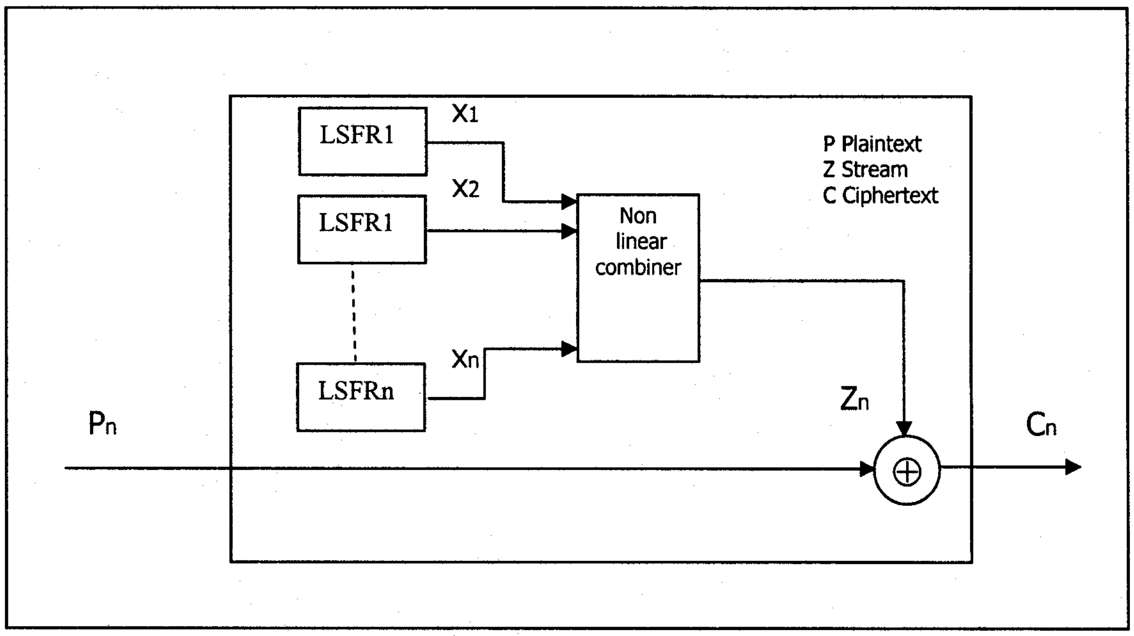

Figure 1.4: LFSR-based stream cipher.

1.4.4 Block Ciphers

A block cipher is an encryption scheme which breaks up the plaintext messages to be transmitted into strings called blocks of a fixed length L, then encrypts one block at a time. When encrypting, a block cipher takes L-bit block of plaintext as input, and output a corresponding L-bit block of ciphertext. The exact transformation is controlled by the secret key. Decryption is similar; the decryption algorithm takes an L-bit block of ciphertext together with the secret key, and yields the original L-bit block of plaintext.

Block ciphers can be contrasted with stream ciphers; a stream cipher operates on individual digits one at a time. The distinction between the two types is not always clear-cut; a block cipher, when used in certain modes of operation, acts effectively as a stream cipher (for example, output feedback mode (*OFB*)).

In 1949 Shannon [2] presented the principles of confusion and diffusion. Because these principles are so successful in capturing the essence of the desired attributes of a block cipher, they have become the cornerstone of block cipher design. Confusion is described as "the use of enciphering transformations that complicate the determination of how the statistics of the ciphertext depend on the statistics of the plaintext" [4] or, more briefly, to make the relation between the key and the ciphertext as complex as possible, thereby hiding the statistical features of the plaintext. On the other hand, diffusion spreads the influence of individual plaintext characters over as much of the ciphertext as possible, thereby hiding the statistical features of the plaintext. Methods of achieving good diffusion and confusion are the core of block cipher design. An effective implementation of Shannon's principle of mixing transformation based on the concepts of "confusion" and "diffusion" is the Substitution Permutation Network (*SPNs*).

1.4.4.1 Substitution-Permutation Networks

Feistel [5] and Feistel, Notz, and Smith [6] were the first to suggest that a Substitution Permutation Network (*SPN*) architecture (see Figure 1.5) was a simple effective implementation of Shannon's principle of mixing transformation based on the concepts of "confusion" and "diffusion". Keying the network can be accomplished by XORing the key bits with the data bits before each round of substitution and after the last round, or by choosing a different set of s-boxes for each key. Boolean functions are the main core for designing s-boxes and the characteristics of the s-boxes depend on the properties of Boolean functions used [7].

1.5 Outline of the Thesis

Chapter two gives some of the mathematical background and the necessary definitions required throughout the thesis.

Chapter three gives a review of some constructions for bent and resilient functions.

In chapter four we introduce new restrictions on the Walsh transform as well as a new construction for bent functions.

In chapter five we classify nonlinear maximum order Boolean resilient functions based on the bent-based and linear-based concept and provide some experimental results for the algebraic immunity of Boolean functions.

Finally in chapter six we give a summary of our results and directions for future work.

Some parts of the work presented in chapter 4 were published in [56].

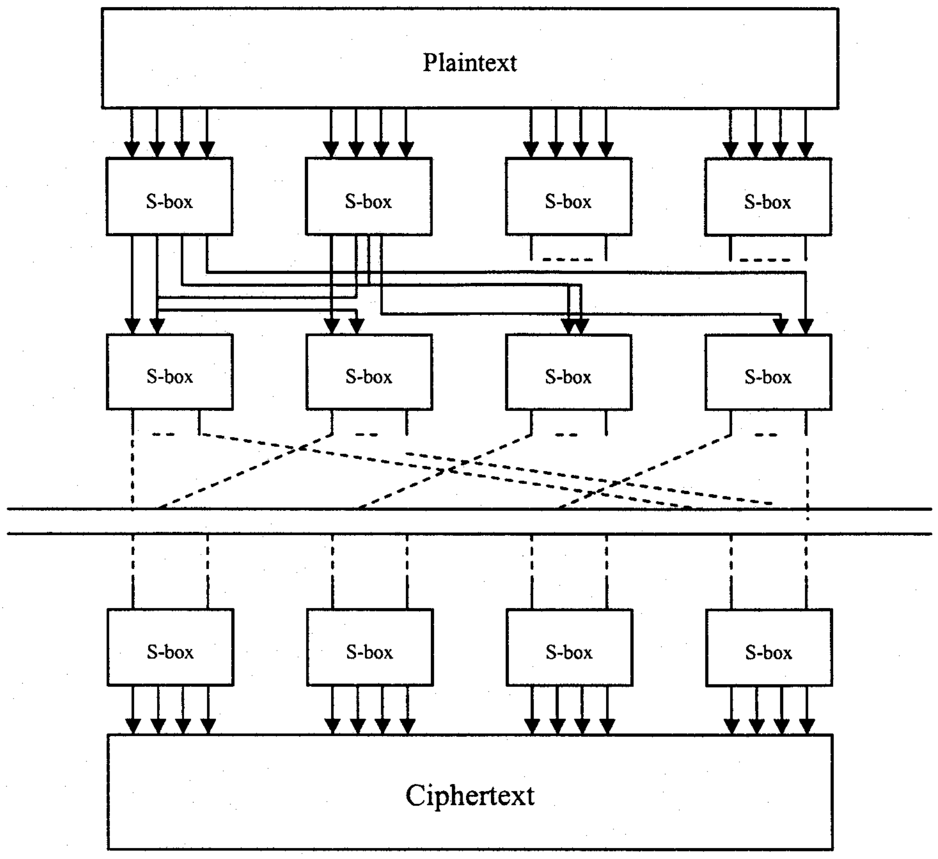

Figure 1.5: Basic substitution permutation network cipher.

Chapter 2

Boolean Functions

2.1 Introduction

A Boolean function is a $\{0,1\}$-valued function defined on the set Z_2^n of all binary words of a given length n. Boolean functions are used in several different types of cryptographic applications, including the design of block ciphers, stream ciphers, and hash function [3].

The most basic representation of a Boolean function is by its binary truth table. The binary truth table of a Boolean function of n variables is denoted $f(x)$ where $f(x) \in \{0,1\}$ and $x = \{x_1, x_2, \ldots, x_n\}, x_i \in \{0,1\}, i = 1, \ldots, n$. The truth table contains 2^n elements corresponding to all possible combinations of the n binary inputs.

Another representation of a Boolean function is over the set $\{1,-1\}$. The polarity truth table of a Boolean function is denoted $\hat{f}(x)$ where $f(x) \in \{0,1\}$ and $\hat{f}(x) = (-1)^{f(x)} = 1 - 2f(x)$. It is also important to note that XOR over

{0,1} is equivalent to real multiplication over {-1,1}. Thus if $h(x) = f(x) \oplus g(x)$ then $\hat{h}(x) = \hat{f}(x).\hat{g}(x)$.

Example: Let $f(x_1, x_2) = x_1 \oplus x_1 x_2$, then it can be represented by the binary truth table as [*f(0,0) f(1,0) f(0,1) f(1,1)*]=[0 1 0 0] and by the polarity truth table as [1 -1 1 1].

Two fundamental properties of Boolean functions are Hamming weight and Hamming distance, discussed as follows.

Hamming weight

The Hamming weight of a Boolean function is the number of ones in the binary truth table. So the Hamming weight of a Boolean function $wt(f)$ is given by

$$wt(f) = \sum_{x \in Z_2^n} f(x) = \frac{1}{2}(2^n - \sum_{x \in Z_2^n} \hat{f}(x)).$$

Hamming distance

The Hamming distance between two Boolean functions $d(f,g)$ is the number of positions in which their truth tables differ. It can be calculated from either the binary truth table or the polarity truth table as follows:

$$d(f,g) = \sum_{x \in Z_2^n} (f(x) \oplus g(x)) = \frac{1}{2}(2^n - \sum_{x \in Z_2^n} \hat{f}(x).\hat{g}(x)).$$

A linear function, $L_\omega(x)$, selected by $\omega \in Z_2^n$ is defined as

$$L_\omega(x) = \omega \cdot x = \omega_1 x_1 \oplus \omega_2 x_2 \oplus .....\omega_n x_n .$$

An affine function is one of the form

$$A_\omega(x) = \omega \cdot x \oplus c ,$$

where $c \in Z_2$.

The Hamming distance to linear functions is an important cryptographic property. Ciphers that employ nearly linear functions can be broken easily by a variety of methods such as linear cryptanalysis [8]. Thus the minimum distance to the set of affine functions is an important indicator of the cryptographic strength of Boolean functions.

2.2 Algebraic Normal Form

The Algebraic Normal Form (*ANF*) describes a Boolean function in terms of an XOR sum of logical AND products of sub-sets of input variables. Any Boolean function *f(x)* of *n* variables admits a unique (*ANF*) and can be written as:

$$f(x) = a_0 \bigoplus_{i=1}^{i=n} a_i x_i \bigoplus_{1 \le i \ne j \le n} a_{ij} x_i x_j \oplus ... \oplus a_{12...n} x_1 x_2 ... x_n .$$

The *ANF* can be derived from the binary truth table in a binary matrix transformation, the Algebraic Normal Form Transformation (*ANFT*). The *ANFT* matrix is its own inverse, so the binary truth table may also be obtained from the *ANF* using the same *ANFT* operation. The most important cryptographic property related to the *ANF* is the algebraic normal form degree of a Boolean function, which is equal to the number of variables in the highest order product term with nonzero coefficient (*ANFD*). We refer to functions of degree two as quadratic, and function of order three as cubic. Affine functions are those Boolean functions of degree at most one.

Example: Let *f(x)*= [0000011110101011], then *f(x)* expressed in its algebraic normal form is given by :

$$f(x) = x_4 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_1x_2x_3 \oplus x_1x_3x_4 \oplus x_2x_3x_4,$$

and its algebraic normal form degree is 3.

Example: Let *f(x)*= [0000111111110000], then *f(x)* expressed in its algebraic normal form is given by :

$$f(x) = x_3 \oplus x_4,$$

and its algebraic normal form degree is 1. Thus f is a linear function.

2.3 Walsh-Hadamard Transform (WHT)

The Walsh-Hadamard Transform expresses a Boolean function in terms of its correlation with all linear functions. Several important cryptographic properties are expressed directly in terms of Walsh transform values.

The *WHT* of a Boolean function is calculated from the polarity truth table as:

$$F(\omega) = \sum_{x \in Z_2^n} (-1)^{f(x)+\omega.x}.$$

The Walsh transform is sometimes called the spectral distribution or the spectrum of a Boolean function.

Example: Let *f(x)*= [0000011100101111], then the spectrum of the Boolean function f is given by

$$F(\omega) = [0\ 0\ 4\ 4\ 12\ \text{-}4\ 0\ 0\ 4\ 4\ 0\ 0\ 0\ 0\ \text{-}4\ \text{-}4].$$

2.4 Affine Equivalence Classes

An affine transform is a mapping from function f to g such that $g(x) = f(Ax \oplus b) \oplus cx \oplus d$, where A is an $n \times n$ invertible matrix, b, c and d are n-bit vector [9]. The affine transformation preserves the nonlinearity and the algebraic degree [10]. Boolean functions that are related by affine transform can be grouped together to form an equivalence class. An affine transformation provides a method of grouping "like" Boolean functions into classes. Thus, we may refer to two Boolean functions as equivalent if they are related by an affine transform. Boolean functions in the same class have similar cryptographic properties. Equivalence classes under affine transformation are of great importance because they provide a reduced practically useful view of the Boolean space. Several cryptographic properties of Boolean functions remain unchanged under an affine transformation. We need only one example from any given class to find all the other Boolean functions in that class. Tables 2.1 and Table 2.2 show the equivalent classes for n =3, 4 variables respectively [9].

class	example	ANFD	NL	| WHT | distribution
1	`0xaa`	1	0	{(0,7),(8,1)}
2	`0xab`	2	1	{(2,7),(6,1)}
3	`0xac`	3	2	{(0,4),(4,4)}

Table 2.1: Equivalent classes for *n*=3.

class	example	ANFD	NL	abs. WHT distribution
1	`0xaa55`	1	0	{(0,15),(16,1)}
2	`0xab55`	4	1	{(2,5),(14,1)}
3	`0xbb55`	3	2	{(0,8),(4,7),(12,1)}
4	`0xaba5`	4	3	{(2,12),(6,3),(10,1)}
5	`0xaaff`	2	4	{(0,12),(8,4)}
6	`0xaba4`	3	4	{(0,6),(4,8),(8,2)}
7	`0xab12`	4	5	{(2,10),(6,6)}
8	`0xac90`	2	6	{(4,16)}

Table 2.2: Equivalent classes for *n*=4.

Boolean functions of 5 variables were classified in [11] into 48 equivalent classes.

Boolean functions of 6 variables were first classified in 1991 by Maiorana [12] where 150357 different classes were counted.

2.5 Cryptographic Criteria for Boolean Functions

1- Balancedness

For cryptographic Boolean functions, it is usually desired that there are equal number of 0's and 1's in the binary truth table. When this is the case, the function is said to be balanced.

Example: Let *f(x)*=[00011110]. Since the number of 0's in *f* is equal to the number of 1's, then *f(x)* is said to be balanced.

2- Nonlinearity (*NL*)

The nonlinearity (*NL*) of a Boolean function is defined as the minimum Hamming distance between f and the set of affine functions [13]. Note that complementing a Boolean function's binary truth table will not change the nonlinearity, so the magnitude of the correlation to all linear functions, of which there are 2^n, is to be considered. The Hamming distance between a pair of functions can be determined by evaluating both functions for all inputs and counting the disagreements. This process has complexity $O(2^n)$. It follows that determining the nonlinearity in this naive fashion will require $O(2^{2n})$ function evaluations, which is infeasible even for small n.

Example: Let n=2 , *f(x)*= x_1x_2 and $a_i \in Z_2$. Then any affine function can be expressed as

$$A_i(x) = a_0 \oplus a_1x_1 \oplus a_2x_2 .$$

By taking all the combinations of $a_i's$, we can generate all the affine functions for n=2 and they are represented in Table 2.3.

To find the nonlinearity of f we calculate the distance between f and all affine functions that are presented in Table 2.3. The minimum Hamming distance is the *NL* of f.

	Affine functions							
f	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
0	0	0	0	0	1	1	1	1
0	0	1	0	1	1	0	1	0
0	0	0	1	1	1	1	0	0
1	0	1	1	0	1	0	0	1
d(*f*,Ai)	1	1	1	3	3	3	3	1

Table 2.3: Distance between *f* and all affine functions.

$$d_{min}=1 \Rightarrow NL=1.$$

The nonlinearity of *f* can be obtained from the Walsh transform of *f* as follows:

$$NL = \frac{1}{2}(2^n - \max_{\omega \in Z_2^n} |F(\omega)|).$$

Using the fast Walsh transform the complexity of calculating the nonlinearity is reduced to $O(n2^n)$. Clearly in order to increase the nonlinearity, $\max_{\omega \in Z_2^n} F(\omega)$ should be decreased. Note that a function is uncorrelated with linear function $L_\omega(x)$ when $F(\omega) = 0$. For cryptography, it would be desirable to find Boolean functions which have all *WHT* values equal to zero, since such functions have no correlation to any affine functions. However, it is known [14] that such functions do not exist. A well known theorem, widely attributed to Parseval [15], states that the sum of the squares of the *WHT* values is the same constant for every Boolean function,

$$\sum_{\omega \in Z_2^n} F^2(\omega) = 2^{2n}.$$

Thus a tradeoff exists in minimizing affine correlation. When a function is altered so that its correlation to some affine function is reduced, the correlation to some other affine function is increased.

Example: Let *f* = [0001]. Then the spectrum of the Boolean function is given by [2 2 2 -2]. Thus $\max_{\omega \in Z_2^n} F(\omega) = 2$ and hence the *NL* is equal to 1.

Example: Let *f* = [1111]. Then the spectrum of the Boolean function is given by [4000]. Thus $\max_{\omega \in Z_2^n} F(\omega) = 4$ and hence the *NL* is equal to 0.

3- Correlation Immunity (*CI*)

A Boolean function is said to be correlation immune of order *m* if the distribution probability of its output is unaltered when any *m* of its input are fixed [16].

Example : Consider the linear function $f(x) = x_1 \oplus x_2 \oplus x_3$

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

This is a 2-resilient function. For example, the set of arguments $[x_1 x_2 x_3]$ giving *f*(x) = 0 contain each possible value of x_i and $x_j, i \neq j$ exactly once. Similarly, the set of arguments $[x_1 x_2 x_3]$ giving *f*(x) = 1 contain each possible value of x_i and $x_j, i \neq j$ exactly once. Hence, when x_i and x_j are fixed to any values, *f*(x) will equal to 1 with probability 1/2 when x_k, $(k \neq i, k \neq j)$ is chosen by coin tossing.

4- Resiliency

A Boolean function is said to be resilient of order *m* [15] if it is correlation immune of order *m* and it is also balanced.

5-Output Bit Independence Criterion (*BIC*)

A Boolean function is said to satisfy the output Bit Independence Criterion if whenever one bit of the input is complemented, the correlation coefficient between every two output bits changes is zero [17].

6-Completeness

A Boolean function is said to satisfy the completeness criterion if every output bit of the function depends on all input arguments [18].

7-Strict Avalanche Criterion (*SAC*)

A Boolean function is said to satisfy *SAC* criteria if whenever a single input bit is complemented, the output bit changes with a probability of one half [17].

8-Higher Order SAC

A Boolean function is said to satisfy higher order *SAC* criteria of order k if any function obtained from f, by keeping k of its input bits constant, satisfies *SAC* [19]. Higher order *SAC* was defined in a different way by Adams and Tavars [20] as follows: A Boolean function is said to satisfy higher order *SAC* of degree k if f changes with a probability of one half whenever i $(1 \le i \le k)$ bits of x are complemented.

9-Propagation Criterion (*PC*)

A Boolean function is said to satisfy Propagation Criterion of degree k (denoted *PC-k*) if f changes with a probability of one half whenever i $(1 \le i \le k)$ bits of x are complemented . Note that this is identical to Adams and Tavars higher order SAC and was defined independently by Preneel [21].

A Boolean function is said to satisfy the extended Propagation Criterion of degree k and order t (*PC-k* order t) if any function obtained from f by keeping t bits constant satisfies *PC-k* [21].

10-Autocorrelation (*AC*)

The autocorrelation transformation of a Boolean function f is given by

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

where $s \in Z_2^n$.

The maximum absolute value in the autocorrelation spectra of f is given by

$$C_{AC} = \max_{s \ne 0} |\hat{r}_f(s)| = \max_{s \ne 0} \sum_x \hat{f}(x)\hat{f}(x \oplus s).$$

Note that low values of C_{AC} are considered good. Maximal values are serious weakness, called the linear structure [22]. Bent functions have the minimum autocorrelation, so they optimize this property.

The above mentioned parameters are of great importance in designing cryptographic Boolean functions. For example the Boolean function to be used in stream cipher systems should satisfy several properties (e.g., balancedness, high nonlinearity, high algebraic degree, high algebraic immunity and high order of correlation immunity). These parameters are important for resisting different kinds of attacks. There is a certain trade-off involved among these parameters. From this, several interesting questions may arise. For example, can the maximum nonlinearity be achieved for a fixed order of correlation immunity and algebraic degree. A series of papers have discussed these problems but the best trade–off among all these parameters is still an open question in designing Boolean functions.

Chapter 3

Bent and Resilient Functions

3.1 Introduction

Bent functions, is an important class of cryptographic Boolean functions. It was defined and first analyzed by Rothaus [23]. He showed that binary bent functions exist only when the dimension n of the vector space Z_2^n is even. Several properties of bent functions were noted by Rothaus including a characterization in terms of Hadamard matrices. Two large classes of bent functions were also presented in his paper. Further properties and constructions and equivalence bounds for bent functions can be found in [24], [25], [26], [27]. Kumer, Scholtz and Welsh [28] defined and studied bent functions from Z_q^n to Z_q. Bent functions have been the subject of great interest in several areas including cryptography. In fact, the Canadian government block cipher standard (CAST [29]) is designed using these functions.

Another important class of Boolean functions for cryptography is that of resilient functions. These functions play a central role in stream cipher design. In the standard model of these ciphers the output of several independent Linear Feedback Register

(LFSR) sequences are combined using a nonlinear Boolean function to produce the keystream. This keystream is bitwise XORed with the message bitstream to produce the cipher. In 1984 Siegenthaler [16] pointed out that if the combining function is not chosen properly, then the whole system is susceptible to a divide-and-conquer attack. He introduced the concept of m-th order correlation immunity for combining functions as a measure of their resistance against such correlation attacks. He also showed that for an n-variable function, of degree *ANFD* and order of correlation immunity m, the following holds:

$$m + ANFD \leq n .$$

Further if the function is balanced then

$$m + ANFD \leq n - 1 .$$

Later on, Guo-Zehn and Massey [30] introduced an equivalent definition of resilient functions using the Walsh transform of the Boolean function by the following equation:

$$F(\omega) = 0 \qquad 0 \leq wt(\omega) \leq m .$$

Example: Let f = [00111100]

ω	0	1	2	3	4	5	6	7
$wt(f)$	0	1	1	2	1	2	2	3
$F(\omega)$	0	0	0	0	0	0	-8	0

Table 3.1: Walsh Transform distribution.

From Table 3.1, we can see that *F(w)* of Hamming weight 1 is zero and *F(0)* is also zero, then the function is resilient with resiliency degree 1.

3.2 Some Properties of Bent Function

1) A Boolean function f is called bent if all the Walsh transform coefficients have the same absolute value, i.e., $|F(\omega)|$ is constant for all ω. By using Parseval's theorem, f is a bent function if and only if $|F(\omega)| = 2^{\frac{n}{2}}$ for all ω and since $|F(\omega)|$ is an integer then n should be even.

2) Bent functions achieve the maximum possible nonlinearity. The nonlinearity of any bent function is given by

$$NL = (2^{n-1} - 2^{\frac{n}{2}-1}).$$

This means that the Hamming distance of f to every affine function is maximum and equal to $(2^{n-1} \pm 2^{\frac{n}{2}-1})$.

3) Bent functions are never balanced. However, for very large n, they become statistically indistinguishable from balanced functions.

4) The order (algebraic degree) of bent functions is at least 2 and not more than $\frac{n}{2}$, i.e., $2 \le ANFD \le \frac{n}{2}$.

Bent functions of higher algebraic degree are preferred from cryptographic point view

5) f is bent if all its derivatives

$$D_s f(x) = f(x) \oplus f(x+s)$$

are balanced, where s is any non zero vector in Z_2^n [23].

5) All the bent functions have zero autocorrelation for all non-zero s in Z_2^n, i.e.,

$$\hat{r}_f(s) = 0,$$

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

3.3 Classification of Bent Functions

Rothaus [23] has classified the 6 variables bent functions with algebraic degree 3 into three classes. For degree 2, we have one class, so the total number of classes is 4. The number of bent functions in 6 variables is discussed in [24], [31] , [32]. Adams and Tavars [24] defined two general classes (i) bent-based bent sequences (BBBS's) and (ii) linear-based bent sequences (LBBS's). For 4 variables, they found that the number of BBBS's is 512 and the number of LBBS's is 384. For 6 variables, they estimated 48201728 as the number of bent function in which 37879808 are BBBS's and 10321920 are LBBS's. In [31], the number 49774592 was estimated as the lower bound for 6 variables bent functions and in [32] Wang introduced an upper bound. In [33] the exact number of all 6-variable bent functions was determined to be equal to 128 x 42386176.

A heuristic algorithm is presented in [34] that efficiently generates Boolean bent functions, which have desirable cryptographic properties including maximum nonlinearity. This algorithm used the algebraic normal forms to generate new bent functions. The algorithm determines new classes of bent functions for up to 14 inputs. The values in Table 3.2 are the number of distinct classes N_C found so far and they are considered as lower bounds for the number of distinct classes for each n [34].

n	8	8	8	10	10	10	10	12	12	12	12	12	14	14	14	14	14	14
ANFD	2	3	4	2	3	4	5	2	3	4	5	6	2	3	4	5	6	7
N_C	1	3	4	1	7	11	27	1	7	20	32	56	1	9	26	43	71	39

Table 3.2: Lower bound on the number of bent function classes [34].

3.4 A Brief Summary of Bent Function Constructions

3.4.1 Rothaus Construction

Rothaus [23] identified two large general classes of bent functions. These constructions were the first non-trivial construction of bent functions introduced in literature since the functions produced can have degrees near *n/2*. It is still one of the most interesting known constructions nowadays. These two classes are given as follows:

a) Let $x, y \in V_n$ and $P(x)$ be an arbitrary polynomial on V_n, then the polynomial $f(x,y)$ on V_{2n} given by

$$f(x,y) = x_1 y_1 \oplus x_2 y_2 \oplus ...... \oplus x_n y_n \oplus P(x),$$

is a bent function.

b) Let $A(x)$, $B(x)$ and $C(x)$ be bent polynomials on V_{2n} such that $A(x) \oplus B(x) \oplus C(x)$ are also bent. Also, let $y, z \in Z_2^n$. Then the polynomial

$$f(x,y,z) = A(x)B(x) \oplus B(x)C(x) \oplus A(x)C(x) \oplus [A(x) \oplus B(x)]y \oplus [A(x) \oplus C(x)]z \oplus yz$$

is bent function on V_{2n+2}.

3.4.2 Maiorana-McFarland Construction

For n even, Maiorana-McFarland [35], [36] class is the set of all bent functions on $Z_2^n = \{(x,y), x, y \in Z_2^{n/2}\}$ of the form

$$f(x,y) = x.\pi(y) \oplus g(y),$$

where π is any permutation on $Z_2^{n/2}$ and g is any Boolean function on $Z_2^{n/2}$.

Other types of constructions are presented by Yarlagadda and Hershey [26] and Dillon [35] called the effective partial-spread construction.

3.5 *A Brief Summary of Resilient Function Constructions*

3.5.1 Siegenthaler's Construction

Let $f_1(x)$ and $f_2(x)$ be an m-th order correlation immune functions of n binary variables such that $P(f_1(x) = 1) = P(f_2(x) = 0) = p$, then the binary-valued function f of n+1 binary random variables defined by the *GF(2)*expression

$$f(x_1, x_2, \ldots, x_{n+1}) = x_{n+1} f_1(x_1, x_2, \ldots, x_n) \oplus (x_{n+1} \oplus 1) f_2(x_1, x_2, \ldots, x_n),$$

is also m-th order correlation immune [16].

3.5.2 Tarannikov's Construction

In [37], Tarannikov introduced the following construction for resilient functions. If we let f be any Boolean function on Z_2^n, then the Boolean function g on Z_2^{n+1} is defined by

$$g(x_1, x_2, \ldots, x_{n+1}) = x_{n+1} f\ (x_1, x_2, \ldots, x_{n-1}, x_n \oplus x_{n+1}).$$

The Walsh transform of g is $F_g(\omega_1, \ldots, \omega_{n+1})$ and is given by

$$\sum_{x \in Z_2^{n'}} (-1)^{\omega.x \oplus f(x_1, \ldots, x_n) \oplus \omega_n.x_n \oplus (\omega_n \oplus \omega_{n+1} \oplus 1).x_{n+1}},$$

and

$$F_g(\omega_1, \ldots, \omega_{n+1}) = 0.$$

If $\omega_n = \omega_{n+1} \oplus 1$, then

$$F_g(\omega_1, \ldots, \omega_{n+1}) = 2F_f(\omega_1, \ldots, \omega_n).$$

Thus $NL_g = 2NL_f$ and if f is m-resilient then g is m-resilient. If $F_f(\omega_1, \ldots, \omega_{n-1}, 1)$ is null for every vector $(\omega_1, \ldots, \omega_{n-1})$ of weight at most m, then g is $(m+1)$-resilient.

Other types of constructions are presented in [38] where the authors introduced a construction of resilient functions based on the idea of a construction of bent function due to Maiorana-McFarland construction [35], [36]. This construction does not permit to design functions with optimum degrees and nonlinearity (see [39], [40]) except for small values of n. Modifications and generalization of this construction have been proposed in [41] and [42].

3.6 Nonlinearity of Resilient Functions

The maximum possible nonlinearity of m-resilient Boolean functions was discussed in many papers. It is well-known that the nonlinearity of a Boolean function doesn't exceed $2^{n-1}+2^{\frac{n}{2}-1}$ [23]. Let $\hat{N}L(n)$ denote the maximum possible nonlinearity of an n-variable function. Also denote the nonlinearity of an n-variable, m-resilient function by $NL(n,m)$. Then for even n bent functions achieve the maximum nonlinearity $(2^{n-1}-2^{\frac{n}{2}-1})$. For odd n, and $n \le 7$, $\hat{N}L(n) = 2^{n-1}-2^{\frac{(n-1)}{2}}$. For some small values of m and n the exact values of maximal nonlinearity are known. For higher values of n, Palash and Sarkar [43] found a non trivial upper bound on the nonlinearity of resilient functions. Their work is presented by the following theorem:

Lemma1: If $n \ge 3$ and $m \le n-3$ then the Walsh values of m-th order resilient function on n variables satisfy $|F(\omega)| \equiv 0 \bmod 2^{m+2}$ [43].

Using lemma 1, it is possible to obtain an upper bound on the nonlinearity of an n-variable, m-resilient function represented by the following theorem.

.

Theorem 1 [43].

1) If n is even and $m+1 > \frac{n}{2}-1$, then $NL(n,m) \le 2^{n-1}-2^{m+1}$.

2) If n is even and $m+1 \le \frac{n}{2}-1$, then $NL(n,m) \le 2^{n-1}-2^{\frac{n}{2}-1}-2^{m+1}$.

3) If n is odd and $2^{m+1} > 2^{n-1}-\hat{N}L(n)$, then $NL(n,m) \le 2^{n-1}-2^{m+1}$.

4) If n is odd and $2^{m+1} > 2^{n-1}-\hat{N}L(n)$, then $NL(n,m)$ is the highest multiple of 2^{n-1} which is $\le \hat{N}L(n)$.

Before we proceed, we would like to introduce few notations for future convenience. By an (*n*, *m*, *ANFD*, *NL*) function we mean an *n*-variable , *m*-resilient Boolean function with algebraic normal form degree *ANFD* and nonlinearity *NL*. By [*n, m, ANFD, NL*] we denote an unbalanced function with the same notation as above. Any component is replaced by '-' if we do not specify it, e.g., (*n, m,-, NL*) if we do not wish to specify the algebraic degree.

Table 3.3 represents the upper bound on $NL(n,m)$ given by theorem 1.

m \ n	0	1	2	3	4	5	6	7	8
5	12	12	8	0					
6	26	24	24	16	0				
7	56	56	56(1)	48	32	0			
8	118(2)	116(3)	112	112(4)	96	64	0		
9	244(5)	244(5)	240	240(6)	224(4)	192	128	0	
10	494(7)	492(8)	488(8)	480	480(1)	448	384	256	0

Table 3.3: The upper bound on nonlinearity given by theorem 1.

(1) An algorithm to construct (7, 2, 2, 56) and (10, 4, 5, 480) functions has been presented in [10].

(2) This function is not achieved yet: (8, 0, 7, 118).

(3) Computer search has yielded to (8, 1, 6, 116) [16].

(4) Computer search has yielded to (8, 3, 4, 112) and (9,4,4,224) [43].

(5) The existence of (9, 0, -, 244), (9, 0, -, 242), (9, 1, -, 244) functions linked to the

question of whether nonlinearity of more than 240 is exist or not [44].

(6) Computer search has yielded to [9, 3, 5, 240] [45].

(7) An algorithm to construct (10, 0, -, 492) functions has been presented in [41].

(8)Computer search has yielded to (10, 1, 8, 488) [46]. Using the weight divisibility results of resilient function involving the algebraic degree, it can be shown that the functions (10, 1, -, 492), (10, 2, -, 488) if all exist are in the form of (10, 1, 8, 492), (10, 2, 7, 488) [47], [48].

Chapter 4

Restrictions on Partial Sum of WHT and Some Bent Functions Constructions

4.1 Introduction

In this chapter, we present some restrictions on the partial sum of the Walsh transform of binary functions and we extend some of the obtained results to functions defined over *GF*(*p*). We also present some simple constructions for binary bent functions of length 2^{2k} using a known bent function of length 2^{2k-2}. Although some of the bent functions obtained using our techniques may belong to some already known classes of bent functions, our construction techniques are simple and can be performed using hand computations starting with the set of 8 bent functions for $n = 2$ (This set consists of all 8 binary vectors of length 4 and Hamming weight 1or 3).

4.2 *Restrictions on the Partial Sum of the Walsh Transform*

Theorem 1. Let F(w) denote the Walsh transform of a Boolean function *f*. Then for any non zero $a \in Z_2^n$ and $b \in Z_2$ we have

$$\sum_{\omega | a.\omega = b} F(\omega) \in \{0, \pm 2^n\}.$$

Proof.

$$\sum_{\omega | a.\omega = b} F(\omega) = \sum_{\omega | a.\omega = b} \sum_{x \in Z_2^n} (-1)^{f(x)} (-1)^{\omega.x}$$

$$= \sum_{\omega | a.\omega = b} (-1)^{\omega.x} \sum_{x \in Z_2^n} (-1)^{f(x)}.$$

Let $l_x(\omega) = \omega.x$, $\omega \in Z_2^n \mid \omega.a = b$ corresponds to a linear function with *n-1* inputs. This function is balanced for all choices of x except for $x = 0$ where we have $l_0(\omega) = 0$ and $x = a$ where we have $l_0(\omega) = (-1)^b$. Hence we have

$$\sum_{\omega | a.\omega = b} F(\omega) = \begin{cases} 2^{n-1}, & x = 0, \\ (-1)^b 2^{n-1}, & x = a, \\ 0, & \mathit{otherwise}. \end{cases}$$

Thus

$$\sum_{\omega | a.\omega = b} F(\omega) = 2^{n-1}((-1)^{f(0)} + (-1)^b (-1)^{f(a)}).$$

The theorem follows by noting that $((-1)^{f(0)} + (-1)^b (-1)^{f(a)}) \in \{0, \pm 2^n\}$. A special case of the above theorem is for a =(100.....00) which implies that the summation of either the first half or the second half of the Walsh transform should be zero.

It is also clear (from the definition of the inverse Walsh transform) that

$$\sum_{\omega \in Z_2^n} F(\omega) \in \{\pm 2^n\}.$$

Thus it means that if the summation of the first half is zero then the summation of the second half is $\pm 2^n$ and vice versa.

Example: Let f: $Z_2^3 \rightarrow Z_n$ be given by f [11111-1-1-1]. Figure 4.1 and Figure 4.2 show the partial sum of the WHT for a=100 and a =001 respectively.

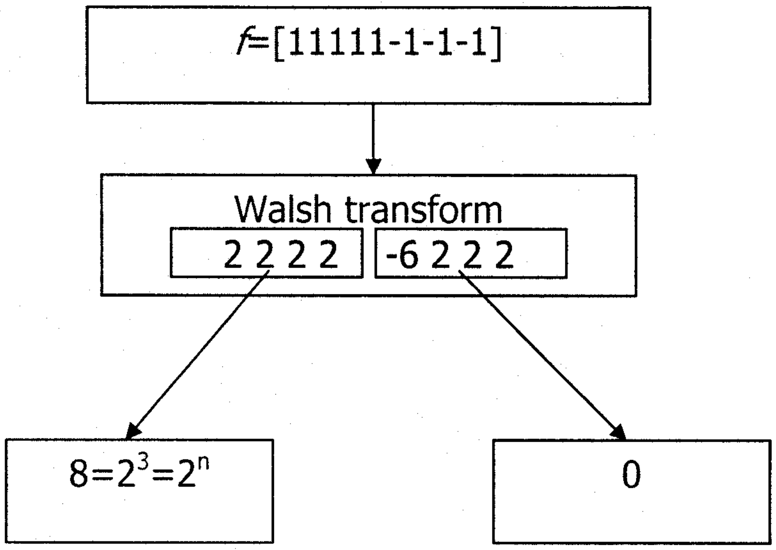

Figure 4.1: Partial sum constrains for a=(100).

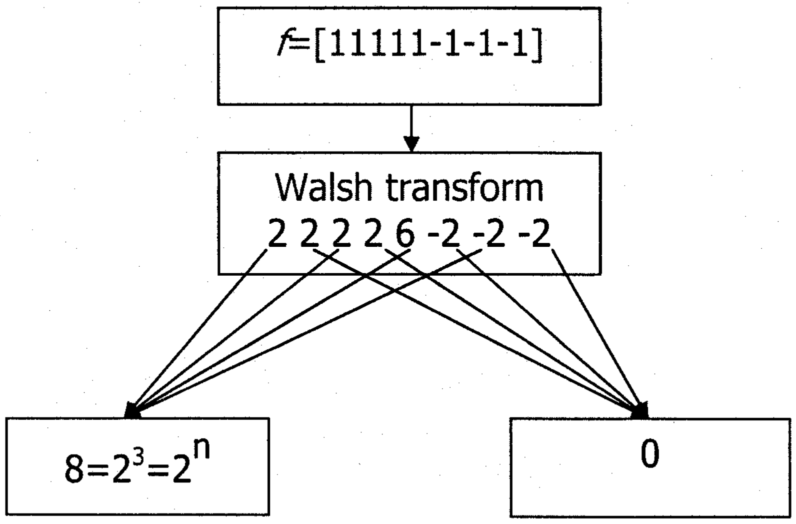

Figure 4.2: Partial sum constrains for a=(001).

Theorem 2. Let $\omega' \in Z_2^n$ be the vector obtained from ω by fixing k of its coordinates. Then we have

$$\sum_{\omega'} F(\omega) \in \{0, S\},$$

where $S = \{\pm 2^n, \pm 2^{n-1}, \ldots\ldots, \pm 2^{n-k-1}\}$.

Proof.

$$\sum_{\omega'} F(\omega') = \sum_{\omega'} \sum_{x \in Z_2^n} (-1)^{f(x)} (-1)^{\omega' . x}$$

$$= \sum_{x \in Z_2^{n'}} (-1)^{f(x)} \sum_{\omega'} (-1)^{\omega' . x}.$$

Note that

$$\sum_{x \in Z_2^{n'}} (-1)^{\omega'.x} = 2^{n-k} \delta(x = x'),$$

where x' is obtained from x by setting all the non-fixed (n - k) coordinates of ω to zero. For example, if $n = 4$; $k = 2$ and $\omega' = (0\,\omega_1\,\omega_2 1)$ then $x' \in \{x_0 00 x_3\}, x_0, x_2 \in Z_2$. Thus

$$\sum_{\omega'} F(\omega') = 2^{n-k} \sum_{x'} (-1)^{f(x')} = 2^{n-k} . \underbrace{(\pm 1 \pm 1 .... \pm 1)}_{2^k},$$

which completes the proof.

A special case of above the theorem is obtained if we set $\omega' = (\omega_0, \omega_1, \omega_2, ..., \omega_{n-2}, b),\ b \in \{0,1\}$. Which again implies that the summation of either the first half or the second half of the Walsh transform is zero.

If we set $\omega' = (\omega_0, \omega_1, \omega_2, ..., \omega_{n-3}, b_0, b_1),\ b_0, b_1 \in \{0,1\}$, then this implies that the summation of each quarter of the Walsh transform $\in \{0, \pm 2^n, \pm 2^{n-1}\}$.

Example: Let $f:\ Z_2^4 \rightarrow Z_n$ be given by f = [11111-1-1-111-11-1-1-11] and let k=2, then $\omega' = (\omega_0, \omega_1, b_0, b_1), b_0, b_1 \in \{0,1\}$. Figure 4.3 shows the *WHT* sum for each quarter.

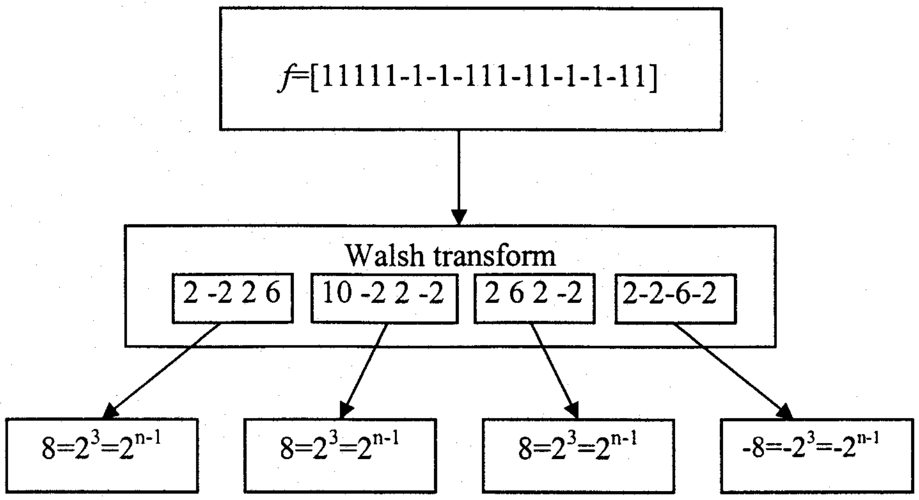

Figure 4.3: Partial sum constrains by fixing b_0 and b_1.

Note that the summation of each quarter $\in \{0, \pm 2^n, \pm 2^{n-1}\}$.

4.2.1 Restrictions on the Partial Sum of the Walsh Transform Over GF(p)

It is interesting to note that the above theorems concerning some partial sums of the Walsh transform can be extended to functions over *GF(p); p* > 2. In this case, the Walsh transform of the function f: $Z_p^n \to Z_P$ is defined by

$$F(\omega) = \sum_{x \in Z_p^n} e^{j\frac{2\pi}{p}(f(x) \oplus \omega . x)},$$

where $\omega \cdot x$ denotes the dot product between ω and x, i.e.

$$\omega \cdot x = \sum_{i=1}^{n} \omega_i x_i .$$

In this case, we show that for any non zero $a \in Z_p^n$ and $b \in Z_P$, we have

$$\sum_{\omega | a \cdot \omega = b} F(\omega) \in p^{n-1} \times \{S\},$$

where $S = \{e^{j\frac{2\pi}{p}f(0)} + e^{j\frac{2\pi}{p}bf(a)} + e^{-j\frac{2\pi}{p}f(-a)}\}$.

The above result follows by noting that

$$\#\{\omega \mid a \cdot \omega = b\} = p^{n-1}$$

and for $a \neq 0$ we have

$$\sum_{\omega | a \cdot \omega = b} e^{j\frac{2\pi}{p}\omega \cdot x} = \begin{cases} p^{n-1}, & x = 0, \\ e^{j\frac{2\pi}{p}b} p^{n-1}, & x = a, \\ e^{-j\frac{2\pi}{p}b} p^{n-1}, & x = -a \bmod p, \\ 0, & \textit{otherwise.} \end{cases}$$

The main reason that the above formula cannot apply to binary functions ($p = 2$) is that $p = 2$ is the only even prime ($a = -a \bmod 2$).

Similarly, Theorem 2 can be extended to the $GF(p)$, $p > 2$ case by noting that

$$\sum_{x \in Z_p^n} e^{j\frac{2\pi}{p}(\omega' \cdot x)} = p^{n-k} \delta(x = x'),$$

where x' is obtained from x by setting all non-fixed $(n - k)$ coordinates of ω to zero.

Thus

$$\sum_{\omega'} F(\omega) = p^{n-k} \sum_{x'} e^{j\frac{2\pi}{p} f(x')} = p^{n-k} \underbrace{(e^{j\frac{2\pi}{p} i_0} + e^{j\frac{2\pi}{p} i_1} + \ldots + e^{-j\frac{2\pi}{p} i_{p^k - 1}})}_{p^k}$$

where $i_j \in Z_p$.

4.3 New Constructions of Bent Functions

4.3.1 Construction 1

Lemma 1. *Let $g(x) = f(Ax \oplus b)$ for any invertible matrix A. Then $NL(g) = NL(f)$.*

Theorem 1. Let $f = [f_1|f_2|f_3|f_4]$: $Z_2^n \rightarrow Z_2$ be a bent function where f_i, i = 1,2,3,4 are binary sequences each of dimension 2^{n-2}. Note that f_i's are not necessarily bent. If we let $\Pi(i, 2, 3, 4) = \{i_1, i_2, i_3, i_4\}$ denote any permutation of the set $\{1,2,3,4\}$, then we can show that the function

$$g = [f_{i_1} | f_{i_2} | f_{i_3} | f_{i_4}]$$

is bent.

Proof: The proof follows by noting that the function g above can be expressed as g(x) = $f(Ax \oplus b)$ where A is an invertible matrix given by

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a_0 & a_1 \\ 0 & 0 & a_2 & a_3 \end{bmatrix} \quad and \quad b = \begin{bmatrix} 0 \\ 0 \\ b_0 \\ b_1 \end{bmatrix}.$$

Note that A is invertible implies that

$$\begin{bmatrix} a_0 & a_1 \\ a_2 & a_3 \end{bmatrix}$$

is also invertible.

Table 4.1 shows the mapping between a_0, a_1, a_2, a_3, b_0, b_1 and $\Pi(1, 2, 3, 4) = \{i_1, i_2, i_3, i_4\}$.

***Example*:** Starting with the bent function $f = [1110]$, and applying the theorem above, we obtain the following set $S = \{[1101]; [1011]; [0111]\}$ of bent functions. It is clear that the functions in $\{f; S\}$ and its complement represent all the bent functions for $n = 2$.

(i_1, i_2, i_3, i_4)	$(a_0, a_1, a_{2,}, a_3, b_0, b_1)$
(0,2,1,3)	(0,1,1,0,0,0)
(0,3,1,2)	(1,1,1,0,0,0)
(0,1,2,3)	(1,0,01,0,0)
(0,1,3,2)	(1,1,0,1,0,0)
(0,3,2,1)	(1,0,1,1,0,0)
(0,2,3,1)	(0,1,1,1,0,0)
(1,3,0,2)	(0,1,1,0,1,0)
(1,2,0,3)	(1,1,1,0,1,0)
(1,0,3,2)	(1,0,0,1,1,0)
(1,0,2 ,3)	(1,1,01,1,0)
(1,2,3,0)	(1,0,1,1,1,0)
(1,3,2,0)	(0,1,1,1,1,0)
(2,0,3,1)	(0,1,1,0,0,1)
(2,1,3,0)	(1,1,1,0,0,1)
(2,3,0,1)	(1,0,0,1,0,1)
(2,3,1,0)	(1,1,0,1,0,1)
(2,1,0,3)	(1,0,1,1,0,1)
(2,0,1,3)	(0,1,1,1,0,1)
(3,1,2,0)	(0,1,1,0,1,1)
(3,0,2,1)	(1,1,1,0,1,1)
(3,2,1,0)	(1,0,0,1,1,1)
(3,2,0,1)	(1,1,0,1,1,1)
(3,0,1,2)	(1,0,1,1,1,1)
(3,1,0,2)	(0,1,1,1,1,1)

Table 4.1: Linear transformations and the corresponding permutations in theorem 1.

***Example*:** Let $\Pi = (2, 1, 3, 4)$. Given the bent function $f = [f_1|f_2|f_3|f_4]$ where

$f1 = [1111100111001010]$,

$f2 = [1001110001010000]$,

$f3 = [0001101100010100]$,

$f4 = [0010010011010100]$,

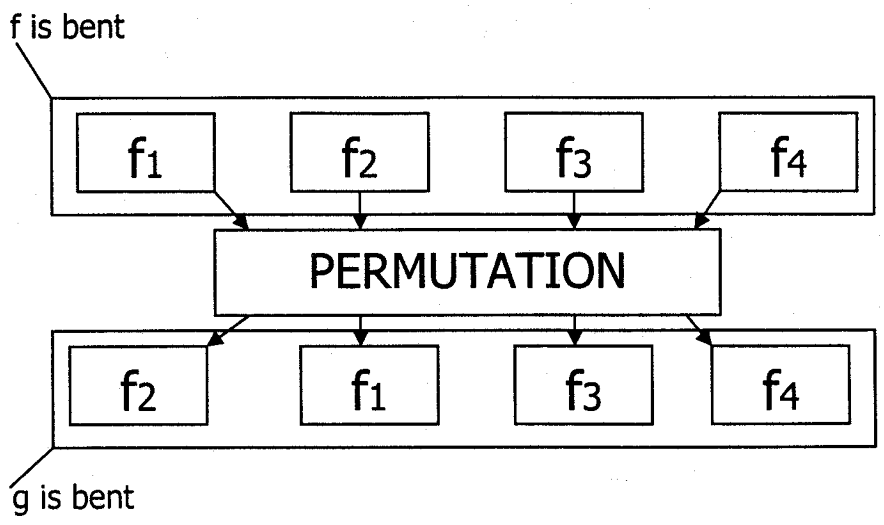

Figure 4.4: Permutation of f_1 ,f_2 ,f_3 and f_4.

then the function g = $[f_2|f_1|f_3|f_4]$ is a bent function with *ANFD* = 3 (see Figure 4.4).

The above theorem can be used to construct bent functions with n + 2 input variables starting with a single input bent function as follows: Let $g_\pi = [f_{\pi(1)} \mid f_{\pi(2)} \mid f_{\pi(3)} \mid f_{\pi(4)}]$, then (see equation (29) in [7]) the function,

$$[f|g|f|g \oplus 1],$$

is bent.

4.3.2 Construction 2

Theorem 2. Let f_1, f_2: $Z_2^n \to Z_2$ be bent functions. Let $f_3(x) = f_1(x \oplus c)$ and $f_4(x) = f_2(x \oplus c)$, where $c \in Z_2^n$. Then the function

$$g = [f_1 \oplus b_1 | f_2 \oplus b_2 | f_2 \oplus b_3 | f_4 \oplus b_4]$$

is bent where one of the b_i's is equal to [11 .... 11] and the other three are equal to [00 .... 00].

Proof

Note that linear function L_n with n input variables can be represented as $[L_{n-1}|(L_{n-1} \oplus b)]$; $b \in \{1,0\}$.

Also note that $d(f(x \oplus c), L)$ is equal to either $d(f(x), L)$ or $2^n - d(f(x), L)$.

$$d(g, L_n) = d([f_1|f_2], L_{n-1}) + d([f_3|f_4], L_{n-1} \oplus b))$$

Without loss of generality, if

$$d([f_1 | f_2], L_{n-1}) = (2^{n-3} - 2^{\frac{n-2}{2}-1}) + (2^{n-3} - 2^{\frac{n-2}{2}-1}) = 2^{n-2} - 2^{\frac{n}{2}-1},$$

then

$$d([f_3 | f_4], L_{n-1} \oplus b) = (2^{n-3} - 2^{\frac{n-2}{2}-1}) + (2^{n-3} + 2^{\frac{n-2}{2}-1}) = 2^{n-2},$$

and hence we have

$$d(g, L_n) = (2^{n-1} - 2^{\frac{n}{2}-1})$$

which means that g is bent.

Example: For f_1=[0010110101110111], f_2=[0001001001000111] and c=[1010], we have f_3=[1110000110111011] and f_4=[1110110101000111], see Table 4.2.

f_1 (x)	$f_3 = f_1(x \oplus c)$	f_2(x)	$f_4 = f_2(x \oplus c)$
0	1	0	0
0	1	0	0
1	1	0	0
0	0	1	1
1	0	0	0
1	0	0	0
0	0	1	1
1	1	0	0
0	1	0	1
1	0	1	0
1	1	0	1
1	1	0	1
0	1	0	1
1	0	1	0
1	1	1	0
1	1	1	0

Table 4.2: Transformation of f_1 and f_2 to f_3 and f_4 respectively.

Then g(x) = [f_1 |f_2|f_3 |$f_4 \oplus 1$], g(x) $\in Z_2^6$

g(x)=[0010110101110111 0001001001000111 1110000110111011 1110110101000111]

is also bent.

The spectral domain dual of this theorem is also satisfied. Let $F_1(\omega)$ and $F_2(\omega)$ be the Walsh spectrum of the two bent functions f_1 and $f_2 \in Z_2^n$ and let $F_3(\omega)= F_1(\omega \oplus c)$ and $F_4(\omega) = F_2(\omega \oplus c)$ where $c \in Z_2^n$. Now, let $\overline{F}_i(\omega) = \frac{F_i(\omega)}{2^{\frac{n}{2}}}$, then

$$G(\omega) = 2^{\frac{n}{2}+1} \times (\overline{F}_1(\omega) \oplus b_1 \mid \overline{F}_2(\omega) \oplus b_2 \mid \overline{F}_3(\omega) \oplus b_3 \mid \overline{F}_4(\omega) \oplus b_4 \mid)$$

is the spectrum of a bent function. In this case, *g* is bent but it is not necessarily that each quarter is bent.

Note that one of the b_i's in the given construction is equal to [11 .... 11] and the other three are equal to [00 .... 00].

Example: Let f_1= [1100 0000 0110 1010] and f_2= [0010 1000 1110 0100] then

$F_1(\omega) =$ [4 -4 -4 4 -4 4 -4 4 4 4 -4 -4 -4 -4 -4 -4],

$F_2(\omega) =$ [4 -4 -4 4 -4 -4 4 4 4 -4 4 -4 4 4 4 4],

$F_3(\omega)= F_1(\omega \oplus c) =$ [4 4 4 4 4 -4 4 -4 -4 -4 4 4 4 -4 -4 4],

$F_4(\omega) = F_2(\omega \oplus c) =$ [-4 -4 -4 -4 4 4 -4 -4 -4 4 -4 4 4 -4 -4 4].

Let b_3=[1111] , then we have

$$G(\omega) = 8 \times [\overline{F}_1(\omega) \mid \overline{F}_2(\omega) \mid \overline{F}_3(\omega) \oplus 1 \mid \overline{F}_4(\omega)] \text{ and}$$

$g(x) = [f_1^{'} \mid f_2^{'} \mid f_3^{'} \mid f_4^{'}]$ is bent, but $f_1^{'}, f_2^{'}, f_3^{'}, f_4^{'}$ are not necessarily bent.

Applying the inverse *WHT* we get

g(x) = [0010000001100100 1101000001101011 1100100011101010 1100011100011010].

It can be easily verified that

$f_1^{'}$ =[0010000001100100], $f_2^{'}$ =[1101000001101011],

$f_3^{'}$ =[1100100011101010] and $f_4^{'}$ = [1100011100011010] are not bent.

Chapter 5

On Some Bent-Based Constructions

5.1 Introduction

Adams and Tavares [24] introduced the classes of bent-based (BB) bent functions and linear-based (LB) bent functions. They conjectured that all bent functions can be classified into one of these classes. Although this conjecture was disproved by Carlet *et al.* [49], bent-based functions still present a very interesting class of cryptographic functions because several fundamental cryptanalytic techniques can be performed by fixing some of the coordinates of the input to the cipher system.

In this chapter, we show that several classes of cryptographic Boolean functions are in fact bent-based functions.

In particular, we show that:

1. All nonlinear resilient functions with maximum resiliency degree, i.e. $(n, n-3, 2, 2^{n-2})$

functions, are either bent-based or linear-based. We provide an explicit count for the functions in both classes.

2. All symmetric bent functions that achieve the maximum possible nonlinearity are bent-based. For *n* even, we have 4 bent-based bent functions. For *n* odd, we also have 4 bent-based functions.

3. There are no bent-based homogeneous functions with degree>2. We also provide a count for linear-based homogeneous functions.

Preliminaries

A sequence *x* of order $2^{k+2,}$ where *k* is a positive integer, is bent-based if it is a concatenation of 2^k bent subsequences of order 4.

Recall that the bent sequences of order 4 are given by

$$\begin{array}{ll} g1 = [1\ 1\ 1 \text{-}\ 1], & -g1 = -[1\ 1\ 1 \text{-}\ 1], \\ g2 = [1\ 1\ \text{-}1\ 1], & -g2 = -[1\ 1\ \text{-}1\ 1], \\ g3 = [1\ \text{-}1\ 1\ 1], & -g3 = -[1\ \text{-}1\ 1\ 1], \\ g4 = [\text{-}1\ 1\ 1\ 1], & -g4 = -[\text{-}1\ 1\ 1\ 1]. \end{array}$$

This set corresponds to the 8 binary vectors of length 4 and nonlinearity 1. In other words, they correspond to binary bent functions with *n*=2. The Walsh transforms for these bent-based functions are given in Table 5.1.

Bent-Based Sequences	**Walsh Transform**
±g1	±(2 2 2 -2)
±g2	±(2 -2 2 2)
±g3	±(2 2 -2 2)
±g4	±(2 -2 -2 -2)

Table 5.1: Walsh transform for *n*=2 bent-based functions.

Note that a sequence x of order $2^{k+2,}$ where k is a positive integer, is linear-based if it is a concatenation of 2^k linear subsequences of order 4. Recall that the linear sequences of order 4 are given by

g1=[1 1 1 1], -g1= -[1 1 1 1],
g2=[-1 1 1 -1], - g2= -[-1 1 1 -1],
g3=[-1 1 -1 1], - g3= -[-1 1- 1 1],
g4=[1 1- 1- 1], - g4= -[1 1 -1- 1].

The Walsh transforms for these linear sequences functions are given in Table 5.2.

Linear Function	**Walsh Transform**
±g1	±(4 0 0 0)
±g2	±(0 0 0 -4)
±g3	±(0 -4 0 0)
±g4	±(0 0 4 0)

Table 5.2: Walsh transforms for n=2 linear-based functions.

Most of the results in this chapter follow by noting the following observations:

1) For n=2, all binary bent functions over *GF(2)* are quadratic. Thus it follows that, for any n>1, all bent-based functions should have the term x_1x_2 in their algebraic normal form representation.

2) For n=1, all bent functions over *GF(p), p>2,* are quadratic. Thus it follows that, for any n>1, all bent-based functions over *GF(p), p>2,* should have the term x_1^2 in their algebraic normal form representation.

3) Let $f(x_1,x_2,...,x_n)=f_1(x_1,x_2,...,x_t)+f_2(x_{t+1},x_{t+2},...,x_n)$ over *GF(p)*, then

$$F(\omega_1,\omega_2,...,\omega_n) = F_1(\omega_1,\omega_2,...,\omega_t)F_2(\omega_{t+1},\omega_{t+2},...,\omega_n).$$

Thus if either f_1 or f_2 is balanced, then f is balanced. This follows by noting that for any balanced function *g*, *G(0)=0*. Since non constant linear functions are balanced, it follows that if f_1 or f_2 is non constant linear function then f is balanced.

5.2 *Nonlinear Resilient Functions with Maximum Resiliency Degree*

In this section, using the results in [38], we show that all resilient functions in the form $(n,n-3,2,2^{n-2})$ are either bent-based or linear-based. Then we extend some of the obtained results to functions defined over *GF*(*p*).

Recall that for an *n*-variable function, of degree *ANFD* and correlation immunity of order m, the following holds:

$$m + ANFD \leq n .$$

Further, if the function is balanced then

$$m + ANFD \leq n - 1 .$$

Thus for nonlinear Boolean functions (i.e., for *ANFD*>1), the maximum resiliency degree *m* is (*n*-3).

Carlet *et al.* [38] proved that $(n,n-3,2,2^{n-2})$ resilient functions can only take the following four forms:

Form 1

$$f(x) = x_i x_j + \sum_{t\in\{[1,n]-(1,2)\}} x_t + ax_i + bx_j + c,$$

$a,b,c \in Z_2$.

Form 2

$$f(x) = (x_i + x_j)(x_i + x_k) + \lambda x + a ,$$

where i, j and k are distinct elements of $[1,n]$ and $a \in Z_2$,

$$\lambda x = \sum_{t=1}^{n} x_t \text{ or } \lambda x = \sum_{t\in\{[1,n]-(i,j)\}} x_t \text{ or } \lambda x = \sum_{t\in\{[1,n]-(i,k)\}} x_t \text{ or } \lambda x = \sum_{t\in\{[1,n]-(j,k)\}} x_t .$$

Form 3

$$f(x) = (x_i + x_j)x_k + \lambda x + a ,$$

where i, j and k are distinct elements of $[1,n]$ and $a \in Z_2$,

$$\lambda x = \sum_{t\in\{[1,n]-i\}} x_t \text{ or } \lambda x = \sum_{t\in\{[1,n]-j\}} x_t \text{ or } \lambda x = \sum_{t\in\{[1,n]-(i,k)\}} x_t \text{ or } \lambda x = \sum_{t\in\{[1,n]-(j,k)\}} x_t .$$

Form 4

$$f(x) = (x_j + x_k)(x_i + x_l) + \lambda x + a ,$$

where i, j and k are distinct elements of $[1,n]$ and $a \in Z_2$,

$$\lambda x = \sum_{t\in\{[1,n]-(i,j)\}} x_t \text{ or } \lambda x = \sum_{t\in\{[1,n]-(i,k)\}} x_t \text{ or } \lambda x = \sum_{t\in\{[1,n]-(l,j)\}} x_t \text{ or } \lambda x = \sum_{t\in\{[1,n]-(l,k)} x_t .$$

In here, based on observation (1) in p 53, we provide a simple restriction on these forms such that the resulting functions are bent-based.

Form 1

$$f(x) = x_1x_2 + \sum_{t\in\{[1,n]-(1,2)\}} x_t + ax_1 + bx_2 + c,$$

$a,b,c \in Z_2$.

By a simple counting argument, it is easy to show that the number of functions described by this form N_{F_1} are given by

$$N_{F_1} = 8.$$

Example: For n=3, we have $N_{F_1} = 8$ and the functions are:

$f(x) = x_1x_2 + x_1 + x_2 + x_3$, $f(x) = x_1x_2 + x_1 + x_2 + x_3 + 1$,

$f(x) = x_1x_2 + x_1 + x_3$, $f(x) = x_1x_2 + x_1 + x_3 + 1$,

$f(x) = x_1x_2 + x_2 + x_3$, $f(x) = x_1x_2 + x_2 + x_3 + 1$,

$f(x) = x_1x_2 + x_3$, $f(x) = x_1x_2 + x_3 + 1$.

Form 2

$$f(x) = (x_1 + x_2)(x_1 + x_i) + \lambda\, x + a,$$

where i, is an element of $[3,n]$ and $a \in Z_2$,

$$\lambda x=\sum_{t=1}^{n} x_t \quad or \quad \lambda x=\sum_{t\in\{[1,n]-(1,i)\}} x_t \quad or \quad \lambda x=\sum_{t\in\{[1,n]-(2,i)\}} x_t \quad or \quad \lambda x=\sum_{t\in\{[1,n]-(1,2)\}} x_t .$$

We have 4 choices for λ, 2 choices for a and *(n-2)* choices for i. Thus the number of functions described by this form, $\mathrm{N}_{\mathrm{F}_2}$, is given by

$$\mathrm{N}_{\mathrm{F}_2}=8(n-2).$$

Example: For n=3, we have $\mathrm{N}_{\mathrm{F}_2}=8$ and the functions are:

$f(x)=(x_1+x_2)(x_1+x_3)+x_1+x_2+x_3$, $f(x)=(x_1+x_2)(x_1+x_3)+x_1+x_2+x_3+1$,

$f(x)=(x_1+x_2)(x_1+x_3)+x_3$, $f(x)=(x_1+x_2)(x_1+x_3)+x_3+1$,

$f(x)=(x_1+x_2)(x_1+x_3)+x_2$, $f(x)=(x_1+x_2)(x_1+x_3)+x_2+1$,

$f(x)=(x_1+x_2)(x_1+x_3)+x_1$, $f(x)=(x_1+x_2)(x_1+x_3)+x_1+1$.

Form 3

$$f(x)=(x_i+x_j)x_k+\lambda x+a,$$

where $i,k\in\{1,2\}$ $i\neq k$, j is an element of $[3, n]$ $a\in Z_2$

$$\lambda x=\sum_{t\in\{[1,n]-i\}} x_t \quad or \quad \lambda x=\sum_{t\in\{[1,n]-j\}} x_t \quad or \quad \lambda x=\sum_{t\in\{[1,n]-(i,k)\}} x_t \quad or \quad \lambda x=\sum_{t\in\{[1,n]-(j,k)\}} x_t .$$

We have 4 choices for λ, 2 choices for a , 2 choices for i and j and *(n-2)* choices for k. Thus the number of functions described by this form, $\mathrm{N}_{\mathrm{F}_3}$, is given by

$$N_{F_3} = 16(n-2).$$

Example: For n=3, we have $N_{F_3} = 16$ and these functions are:

$f(x) = (x_1 + x_3)x_2 + x_1 + x_2$, $\quad f(x) = (x_1 + x_3)x_2 + x_1 + x_2 + 1$,

$f(x) = (x_1 + x_3)x_2 + x_2 + x_3$, $\quad f(x) = (x_1 + x_3)x_2 + x_2 + x_3 + 1$,

$f(x) = (x_1 + x_3)x_2 + x_1$, $\quad f(x) = (x_1 + x_3)x_2 + x_1 + 1$,

$f(x) = (x_1 + x_3)x_2 + x_3$, $\quad f(x) = (x_1 + x_3)x_2 + x_3 + 1$,

$f(x) = (x_2 + x_3)x_1 + x_1 + x_2$, $\quad f(x) = (x_2 + x_3)x_1 + x_1 + x_2 + 1$,

$f(x) = (x_2 + x_3)x_1 + x_1 + x_3$, $\quad f(x) = (x_2 + x_3)x_1 + x_1 + x_3 + 1$,

$f(x) = (x_2 + x_3)x_1 + x_2$, $\quad f(x) = (x_2 + x_3)x_1 + x_2 + 1$,

$f(x) = (x_2 + x_3)x_1 + x_3$, $\quad f(x) = (x_2 + x_3)x_1 + x_3 + 1$.

Form 4

$$f(x) = (x_1 + x_i)(x_2 + x_j) + \lambda x + a ,$$

where i and j are distinct elements of $[3,n]$ and $a \in Z_2$,

$$\lambda x = \sum_{t\in\{[1,n]-(1,2)\}} x_t \quad or \quad \lambda x = \sum_{t\in\{[1,n]-(2,i)\}} x_t \quad or \quad \lambda x = \sum_{t\in\{[1,n]-(1,j)\}} x_t \quad or \quad \lambda x = \sum_{t\in\{[1,n]-(i,j)\}} x_t .$$

We have 4 choices for λ, 2 choices for a and C_2^{n-2} choices for i and j. Thus the number of functions described by this form, $\mathrm{N}_{\mathrm{F}_4}$ is given by

$$\mathrm{N}_{\mathrm{F}_4} = 16 \cdot C_2^{n-2}.$$

Example: For n=3, there are no functions of the form 4. For n=4, we have $\mathrm{N}_{\mathrm{F}_4} = 16$ and these functions are:

$f(x) = (x_1 + x_3)(x_2 + x_4) + x_3 + x_4$,	$f(x) = (x_1 + x_3)(x_2 + x_4) + x_3 + x_4 + 1$,
$f(x) = (x_1 + x_3)(x_2 + x_4) + x_1 + x_4$,	$f(x) = (x_1 + x_3)(x_2 + x_4) + x_1 + x_4 + 1$,
$f(x) = (x_1 + x_3)(x_2 + x_4) + x_2 + x_3$,	$f(x) = (x_1 + x_3)(x_2 + x_4) + x_2 + x_3 + 1$,
$f(x) = (x_1 + x_3)(x_2 + x_4) + x_1 + x_2$,	$f(x) = (x_1 + x_3)(x_2 + x_4) + x_1 + x_2 + 1$,
$f(x) = (x_1 + x_4)(x_2 + x_3) + x_3 + x_4$,	$f(x) = (x_1 + x_4)(x_2 + x_3) + x_3 + x_4 + 1$,
$f(x) = (x_1 + x_4)(x_2 + x_3) + x_1 + x_3$,	$f(x) = (x_1 + x_4)(x_2 + x_3) + x_1 + x_3 + 1$,
$f(x) = (x_1 + x_4)(x_2 + x_3) + x_2 + x_4$,	$f(x) = (x_1 + x_4)(x_2 + x_3) + x_2 + x_4 + 1$,
$f(x) = (x_1 + x_3)(x_2 + x_4) + x_1 + x_2$,	$f(x) = (x_1 + x_3)(x_2 + x_4) + x_1 + x_2 + 1$.

The total number of BB functions are given by

$$\begin{aligned} \mathrm{N}_{\mathrm{BBT}} &= 8(C_0^{n-2} + C_1^{n-2} + 2 \cdot C_1^{n-2} + 2 \cdot C_2^{n-2}) \\ &= 8(1 + n - 2 + 2(n-2) + (n-2)(n-3)) \qquad (5.1) \\ &= 8(n-1)^2. \end{aligned}$$

Table 5.3 presents the number of BB functions for each form for different n.

n \ f	3	4	5	6	7
Form1	8	8	8	8	8
Form2	8	16	24	32	40
Form3	16	32	48	64	80
Form4	0	16	48	96	160
Total	32	72	128	200	288

Table 5.3: The number of BB functions for different n.

Carlet [38] calculated the total number of quadratic functions that have the form $(n, n-3, 2, 2^{n-2})$ by :

$$N_Q = \frac{1}{3}[n(n-1)(3n-2)(n+1)]. \qquad (5.2)$$

Using (5.1) with (5.2) one can obtain the number of LB functions as follows,

$$\begin{aligned} N_{LB} &= N_Q - N_{BBT} \\ &= \{\frac{1}{3}[n(n-1)(3n-2)(n+1)] - 8(n-1)^2\} \\ &= \{(n-1)[\frac{1}{3}n(3n-2)(n+1) - 8(n-1)]\}. \end{aligned}$$

Table 5.4 presents the number of BB functions and LB functions for different value of *n*.

n \ f	3	4	5	6	7
BB	32	72	128	200	288
LB	24	128	392	32	1840
Total	56	200	520	920	2128

Table 5.4: The number of BB and LB functions for different *n*.

5.2.1 Extensions to *GF(p)*

It is interesting to note that the above resilient functions constructions can be extended to functions over *GF(p); p > 2* with the same resilient degree (*n*-3) . In this case, the Walsh transform of the function f: $Z_p^n \rightarrow Z_P$ is defined by

$$F(\omega) = \sum_{x \in Z_p^n} e^{j\frac{2\pi}{p}(f(x)+\omega.x)},$$

where $\omega \cdot x$ denotes the dot product between ω and x over GF(p), i.e.,

$$\omega \cdot x = \sum_{i=1}^{n} \omega_i x_i \mod \text{p}.$$

Unlike the binary case where bent functions exist only for even *n*, bent functions over *GF(p), p>2*, exist for all *n* including *n*=1. Table 5.5 shows all the 18 bent functions for the case *n*=1 and *p*=3.

Bent	100	200	010	110	020	220	001	101	011
Functions	211	121	221	002	202	112	212	022	122

Table 5.5: Bent functions over GF(3), n=1.

All the results obtained in this section are based on observation 2 and 3 in pages 53 and 54. In particular, in order to prove that f is a resilient function of degree n-3, we need to show that any function, f', obtained from f by fixing $t \leq n\text{-}3$ variable is balanced. We do this by showing that we can write f' as the sum of two functions (with disjoint arguments) where one of these functions is a balanced linear function. From observation *3* in page 53, it follows that f is a resilient function of degree n-3 .The following 3 forms are bent-based resilient functions of degree n-3.

Form 1

$$f(x) = ax_1^2 + b \sum_{t\in\{[1,n]-(1,2)\}} x_t + cx_1 + d,$$

where $a,b \in Z_p^* \quad c,d \in Z_p$.

In order to prove that f is a resilient function of degree n-3, we need to show that any function, f', obtained from f by fixing $t \leq n-3$ variable is balanced. Thus we need to consider the following three cases.

Case 1: x_1 is fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \ldots, x_{i_t}$ where $i_l \neq i_k \; for \, l \neq k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = d' + b\sum_{\substack{l=t+1\\ i_l\notin\{1,2\}}}^{n} x_{i_l},$$

where $d' \in Z_p$.

Case 2: x_1 is not fixed i.e., we fix $x_{i_2}, x_{i_3},\dots,x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = ax_1^2 + cx_1 + b\sum_{\substack{l=t+1\\ i_l\notin\{1,2\}}}^{n} x_{i_l} + d.$$

Case 3: $t = 0$, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = ax_1^2 + cx_1 + b\sum_{l=3}^{n} x_l + d\ .$$

In Form 1, we have p-1 choices for a, (p-1) choices for b, p choices for c, and p choices for d. Thus the number of such functions is given by

$$N_1(p) = p^2(p-1)^2.$$

Form 2

$$f(x) = (ax_1 + bx_i)(cx_1 + dx_j) + e\lambda\, x + h,$$

where $a,b,c,d,e \in Z_p^*$ $h \in Z_p$, $\lambda\, x$ can take one of the following values:

$$1)\ \lambda\,\mathrm{x} = \sum_{t\in\{[1,n]-(1,i)\}} \mathrm{x}_t \quad 2)\ \lambda\,\mathrm{x} = \sum_{t\in\{[1,n]-(1,j)\}} \mathrm{x}_t \quad 3)\ \lambda\,\mathrm{x} = \sum_{t\in\{[1,n]-(i,j)\}} \mathrm{x}_t\ .$$

Similar to Form 1, in order to prove that *f* is a resilient function of degree *n*-3, we need to show that any function, f', obtained from *f* by fixing $t \le n-3$ variable is balanced. Thus we need to consider five cases for the forms 2.1, 2.2 and 2.3.

Form 2.1:

Case 1: x_1 is fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \dots, x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}}, \dots, x_{i_n}) = b' x_i x_j + c' x_i + d' x_j + e \sum_{\substack{l=t+1 \\ i_l \notin \{1,i\}}}^{n} x_{i_l} + h',$$

where $b', c', d' \in Z_p^*$ $h' \in Z_p$.

Case 2: $x_1 x_i$ are fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \dots, x_i, \dots, x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$ $i \ne 1$. For this case, we can express *f* as

$$f'(x_{i_{t+1}}, \dots, x_{i_n}) = d' x_j + e \sum_{\substack{l=t+1 \\ i_l \notin \{1,i,j\}}}^{n} x_{i_l} + h'.$$

Case 3: $x_1, x_i x_j$ are fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \dots, x_i, \dots, x_j, \dots, x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$, $i \ne 1$, $j \ne 1$.

For this case, we can express f' as

$$f'(x_{i_{t+1}}, \dots, x_{i_n}) = h' + e \sum_{\substack{l=t+1 \\ i_l \notin \{1,i\}}}^{n} x_{i_l}.$$

Case 4: $x_1, x_i x_j$ are not fixed i.e., we fix $x_{i_2}, x_{i_3}, \dots, x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$, $i \ne 1$, $j \ne 1$.

For this case, we can express f' as

$$f'(x_{i_{t+1}}, \dots, x_{i_n}) = (ax_1 + bx_i)(cx_1 + dx_j) + ex_j + e \sum_{\substack{l=t+1 \\ i_l \notin \{1,i,j\}}}^{n} x_{i_l} + h'.$$

Case 5: $t = 0$, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = (ax_1 + bx_i)(cx_1 + dx_j) + ex_j + e\sum_{\substack{l=2\\ l\notin\{i,j\}}}^{n} x_l + h'.$$

Form 2.2

Case 1: x_1 is fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \dots, x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = b'x_ix_j + c'x_i + d'x_j + e\sum_{\substack{l=t+1\\ i_l\notin\{1,j\}}}^{n} x_{i_l} + h',$$

where $b', c', d' \in Z_p^*$ $\quad h' \in Z_p$.

Case 2: x_1x_i are fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \dots, x_i, \dots, x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$

$i \neq 1$. For this case, we can express f as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = d'x_j + e\sum_{\substack{l=t+1\\ i_l\notin\{1,j\}}}^{n} x_{i_l} + h'.$$

Case 3: x_1, x_ix_j are fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \dots, x_i, \dots, x_j, \dots, x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$, $i \neq 1$, $j \neq 1$. For this case, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = h' + e\sum_{\substack{l=t+1\\ i_l\notin\{1,j\}}}^{n} x_{i_l}.$$

Case 4: x_1, x_ix_j are not fixed i.e., we fix $x_{i_2}, x_{i_3}, \dots, x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$, $i \neq 1$, $j \neq 1$.

For this case, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = (ax_1 + bx_i)(cx_1 + dx_j) + ex_i + e\sum_{\substack{l=t+1\\ i_l\notin\{1,i,j\}}}^{n} x_{i_l} + h'.$$

Case 5: $t = 0$, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = (ax_1 + bx_i)(cx_1 + dx_j) + ex_i + e\sum_{\substack{l=2\\ l\notin\{i,j\}}}^{n} x_l + h'.$$

Form 2.3

Case 1: x_1 is fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \ldots, x_{i_t}$ where $i_j \neq i_k$ *for* $l \neq k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}}, \ldots, x_{i_n}) = b' x_i x_j + c' x_i + d' x_j + e \sum_{\substack{l=t+1 \\ i_l \notin \{i,j\}}}^{n} x_{i_l} + h,.$$

where $b', c', d' \in Z_p^* \quad h' \in Z_p$.

Case 2: $x_1 x_i$ are fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \ldots, x_i, \ldots, x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$

$i \neq 1$. For this case, we can express f as $f'(x_{i_{t+1}}, \ldots, x_{i_n}) = d' x_j + e \sum_{\substack{l=t+1 \\ i_l \notin \{i,j\}}}^{n} x_{i_l} + h'$.

Case 3: $x_1, x_i x_j$ are fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \ldots, x_i, \ldots, x_j, \ldots, x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$, $i \neq 1, j \neq 1$.

For this case, we can express f' as

$$f'(x_{i_{t+1}}, \ldots, x_{i_n}) = h' + e \sum_{\substack{l=t+1 \\ i_l \notin \{i,j\}}}^{n} x_{i_l}.$$

Case 4: $x_1, x_i x_j$ are not fixed i.e., we fix $x_{i_2}, x_{i_3}, \ldots, x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$, $i \neq 1, j \neq 1$.

For this case, we can express f' as

$$f'(x_{i_{t+1}}, \ldots, x_{i_n}) = (ax_1 + bx_i)(cx_1 + dx_j) + \sum_{\substack{l=t+1 \\ i_l \notin \{i,j\}}}^{n} x_{i_l} + h'$$

Case 5: $t = 0$, we can express f' as

$$f'(x_{i_{t+1}}, \ldots, x_{i_n}) = (ax_1 + bx_i)(cx_1 + dx_j) + ex_1 + e \sum_{\substack{l=2 \\ l \notin \{i,j\}}}^{n} x_l + h'.$$

So the forms 2.1, 2.2 and 2.3 are n-3 resilient functions. We have 3 choices for λx, p-1 choices for a, p-1 choices for b, p-1 choices for c, p-1 choices for d, p-1 choices for e, p choices for h and C_2^{n-1} choices for i and j. Thus the number of such functions is given by

$$N_2(p) = 3p(p-1)^5 C_2^{n-1}.$$

Form 3

$$f(x) = (ax_1 + bx_i)x_1 + c\lambda\, x + d,$$

where $a,b,c \in Z_p^*$ $d \in Z_p$, $\lambda\, x$ can take one of the following values:

1) $\lambda\, x = \sum_{t\in\{[1,n]-1\}} x_t$ 2) $\lambda\, x = \sum_{t\in\{[1,n]-i\}} x_t$ 3) $\lambda\, x = \sum_{t\in\{[1,n]-(1,i)\}} x_t$.

In order to prove that *f* is a resilient function of degree *n*-3, we need to show that any function, f', obtained from *f* by fixing $t \le n-3$ variable is balanced. Thus we need to consider four cases for the forms 3.1, 3.2 and 3.3.

Form 3.1

Case 1: x_1 is fixed, i.e., we fix $x_1, x_{i_2}, x_{i_3}, \ldots, x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}},\ldots,x_{i_n}) = b'x_i + c\sum_{\substack{l=t+1\\ i_l\notin\{1,i\}}}^{n} x_{i_l} + d' \ ,$$

where $b' \in Z_p^*$ $d' \in Z_p$.

Case 2: x_1x_i are fixed, i.e., we fix $x_1, x_{i_2}, x_{i_3}, \ldots, x_i, \ldots, x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$ $i \ne 1$. For this case, we can express *f* as

$$f'(x_{i_{t+1}},\ldots,x_{i_n}) = c\sum_{\substack{l=t+1\\ i_l\ne 1}}^{n} x_{i_l} + d'.$$

Case 3: x_1x_i are not fixed, i.e., we fix $x_{i_2}, x_{i_3}, \ldots, x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$ $i \ne 1$. For this case, we can express *f* as

$$f^{'}(x_{i_{t+1}},...,x_{i_n}) = (ax_1 + bx_i)x_1 + cx_i + cx_j + c\sum_{\substack{l=t+1 \\ i_l \notin \{1,i,j\}}}^{n} x_{i_l} + d \ .$$

Case 4: $t = 0$, we can express $f^{'}$ as

$$f^{'}(x_{i_{t+1}},...,x_{i_n}) = (ax_1 + bx_i)x_1 + cx_i + cx_j + c\sum_{\substack{l=2 \\ l \notin \{i,j\}}}^{n} x_l + d \ .$$

Form 3.2

Case 1: x_1 is fixed i.e., we fix $x_1, x_{i_2}, x_{i_3},..., x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$.

For this case, we can express $f^{'}$ as

$$f^{'}(x_{i_{t+1}},...,x_{i_n}) = b^{'}x_i + c\sum_{\substack{l=t+1 \\ i_l \notin \{1,i\}}}^{n} x_{i_l} + d^{'} \ ,$$

where $b^{'} \in Z_p^{*}$ $\quad d^{'} \in Z_p$.

Case 2: $x_1 x_i$ are fixed i.e., we fix $x_1, x_{i_2}, x_{i_3},..., x_i,..., x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$ $i \neq 1$. For this case, we can express *f* as

$$f^{'}(x_{i_{t+1}},...,x_{i_n}) = c\sum_{l=t+1}^{n} x_{i_l} + d^{'} \ .$$

Case 3: $x_1 x_i$ are not fixed i.e., we fix $x_{i_2}, x_{i_3},..., x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$ $\quad i \neq 1$. For this case, we can express *f* as

$$f^{'}(x_{i_{t+1}},...,x_{i_n}) = (ax_1 + bx_i)x_1 + cx_1 + cx_j + c\sum_{\substack{l=t+1 \\ i_l \notin \{1,i,j\}}}^{n} x_{i_l} + d \ .$$

Case 4: $t = 0$, we can express $f^{'}$ as

$$f^{'}(x_{i_{t+1}},...,x_{i_n}) = (ax_1 + bx_i)x_1 + cx_1 + cx_j + c\sum_{\substack{l=2 \\ l \notin \{i,j\}}}^{n} x_l + d \ .$$

Form 3.3

Case 1: x_1 is fixed i.e., we fix $x_1, x_{i_2}, x_{i_3},..., x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = b'x_i + c\sum_{\substack{l=t+1\\ i_l\notin\{1,i\}}}^{n} x_{i_l} + d'.$$

Case 2: x_1x_i are fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, \dots, x_i, \dots, x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$ $i \neq 1$. For this case, we can express *f* as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = c\sum_{\substack{l=t+1\\ i_l\notin\{1\}}}^{n} x_{i_l} + d',$$

where $b' \in Z_p^*$ $\quad d' \in Z_p$.

Case 3: x_1x_i are not fixed i.e., we fix $x_{i_2}, x_{i_3}, \dots, x_{i_t}$ where $i_l \neq i_k$ *for* $l \neq k$ $\quad i \neq 1$. For this case, we can express *f* as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = (ax_1 + bx_i)x_1 + cx_j + c\sum_{\substack{l=t+1\\ i_l\notin\{1,i,j\}}}^{n} x_{i_l} + d.$$

Case 4: $t = 0$, we can express f' as

$$f'(x_{i_{t+1}},\dots,x_{i_n}) = (ax_1 + bx_i)x_1 + cx_j + c\sum_{\substack{l=2\\ l\notin\{i,j\}}}^{n} x_l + d.$$

So the Forms 3.1, 3.2 and 3.3 are *n*-3 resilient functions. We have 3 choices for λx, *p*-1 choices for *a* , *p*-1 choices for *b*, *p*-1 choices for *c*, *p* choices for *d*, and *n*-1choices for *i*. Thus the number of such functions is given by

$$N_3(p) = 3p(p-1)^3(n-1).$$

We obtain also a construction in the form of $(n, n-2, 2, -)$

$$f(x) = ax_1^{\;2} + \lambda x + c,$$

where $a \in Z_p^*$ $c \in Z_p$, $\lambda\, x$ can take one of the following values:

i) $\lambda\, x = \sum\limits_{t\in\{[1,n]\}} \lambda_t x_t$ *ii)* $\lambda\, x = \sum\limits_{t\in\{[1,n]-1\}} \lambda_t x_t$, $\lambda_t \in Z_P^*\{1,2\}$.

In order to prove that *f* is a resilient function of degree *n*-2, we need to show that any function, f', obtained from *f* by fixing $t \le n-2$ variable is balanced. Thus we need to consider the following three cases for the form *i* and *ii*.

Form *i*

Case 1: x_1 is fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, ..., x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}}, ..., x_{i_n}) = c' + \sum_{l=t+1}^{n} \lambda_{i_l} x_{i_l} \ ,$$

where $c' \in Z_p$.

Case 2: x_1 is not fixed i.e., we fix $x_{i_2}, x_{i_3}, ..., x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}}, ..., x_{i_n}) = ax_1^2 + \lambda_1 x_1 + \sum_{\substack{l=t+1 \\ i_l \notin \{1\}}}^{n} \lambda_{i_l} x_{i_l} + c \,.$$

Case 3: $t = 0$, we can express f' as

$$f'(x_{i_{t+1}}, ..., x_{i_n}) = ax_1^2 + \lambda_1 x_1 + \sum_{l=2}^{n} \lambda_l x_l + c \,.$$

Form *ii*

Case 1: x_1 is fixed i.e., we fix $x_1, x_{i_2}, x_{i_3}, ..., x_{i_t}$ where $i_l \ne i_k$ *for* $l \ne k$.

For this case, we can express f' as

$$f'(x_{i_{t+1}}, ..., x_{i_n}) = c' + \sum_{\substack{l=t+1 \\ l \notin \{1\}}}^{n} \lambda_{i_l} x_{i_l} \,,$$

where $c^{'} \in Z_p$.

Case 2: x_1 is not fixed i.e., we fix $x_{i_2}, x_{i_3}, \ldots, x_{i_t}$ where $i_l \neq i_k \; for\, l \neq k$. For this case, we can express $f^{'}$ as

$$f^{'}(x_{i_{t+1}}, \ldots, x_{i_n}) = ax_1^2 + \sum_{\substack{l=t+1 \\ i_l \neq 1}}^{n} \lambda_{i_l} x_{i_l} + c.$$

Case 3: $t = 0$, we can express $f^{'}$ as

$$f^{'}(x_{i_{t+1}}, \ldots, x_{i_n}) = ax_1^2 + \sum_{l=2}^{n} \lambda_l x_l + c.$$

So the forms *i* and *ii* are *n*-2 resilient functions. We have 2 choices for $\lambda\, x$, *p*-1 choices for a, and *p*-1 choices for *b*. Thus the number of such functions is given by

$$N_1(p) = 2p(p-1).$$

5.3 Symmetric Functions

Symmetric Boolean functions have the property that the function value depends only on the value of the Hamming weight of the input [53]. Consequently, the truth table of the symmetric function on Z_2^n can be replaced by a vector v_f of length $n + 1$ where the components $v_f(i)$ for $0 \leq i \leq n$ represent the function value for vectors of weight *i*. The vector v_f is called the value vector of the symmetric Boolean function *f*. The properties of the symmetric Boolean functions such as balancedness, resiliency, algebraic degree and nonlinearity were studied in [50]. Although random symmetric functions do not behave well with respect to a combination of nonlinearity, algebraic degree, and resiliency, are still attractive option for low memory software and in hardware implementations, where only a low number of gates is available [51].

Savicky proved in [52] that all symmetric bent functions are quadratic and the number of bent functions is always four. Since these functions are quadratic, then they are either linear-based or bent based [24], but since in the case of symmetric Boolean function all the terms of the same degree should appear in the *ANF,* then the term x_1x_2 always exists. Hence all the four symmetric bent functions are bent-based bent function.

Example: $f(x) = x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_2x_4 \oplus x_3x_4$. Then *f(x)* is bent and can be represented by its binary truth table as *f(x)*=[0001 0111 0111 1110] which is a BB function .

In the case of odd *n,* Maitra and Sarkar [53] proved that the maximum nonlinearity achievable by n -variable symmetric Boolean function is $2^{n-1} - 2^{\frac{n-1}{2}}$.

They also showed that there are exactly four possible symmetric Boolean functions achieving the nonlinearity $2^{n-1} - 2^{\frac{n-1}{2}}$, and all these functions are quadratic. It is clear that all these 4 functions are also BB functions for the same reason in the case of even *n.*

Example: $f(x) = x_1x_2 \oplus x_1x_3 \oplus x_2x_3$. *f(x)* can be represented by its binary truth table as *f(x)*=[0001 0111]. It is clear that *f* is a BB function with maximum nonlinearity 2.

5.4 Homogeneous Functions

A Boolean function is said to be homogenous if all the terms in its algebraic normal form (*ANF*) have the same degree.

Recall that any Boolean function *f* can be expressed by the following polynomial

$$f(x) = \oplus_{i \in Z_2} a_i x^{(i)},$$

where $x^{(i)} = x_1^{a_1} \ldots . x_n^{a_n}$ and $a_i \in Z_2$. We say that a Boolean function *f(x)* is homogenous of degree *ANFD* if $a_i = 0$ whenever i has weight not equal to *ANFD* , i.e., $a_i = 0 \; if \; wt(i) \neq ANFD.$

For $ANFD \geq 3$ the term which contains the subterm $x_1 x_2$ is multiplied with other terms resulting in a function that is not either LB or BB. Therefore, we cannot obtain BB functions for $ANFD \geq 3$. However, if these terms vanished we will obtain LB functions. From this observation, the number of LB functions is given by

$$N_{LB} = 2^{(C^{n}_{ANFD} - C^{n-2}_{ANFD-2})} - 1,$$

where $2^{C^{n}_{ANFD}}$ represents the total number of monomials of degree *ANFD* , and $2^{C^{n-2}_{ANFD-2}}$ represents the monomial which will result in mixed terms. The "1" term in the expression above is used to exclude the all zero function from the count.

Example : Let *n*=4 and *ANFD* =3, then $f(x) = a_0 x_1 x_2 x_3 + a_1 x_1 x_2 x_4 + a_2 x_2 x_3 x_4 + a_3 x_1 x_3 x_4$.

In this case, the number of LB functions is given by $N_{LB} = 2^{(C^{4}_{ANFD} - C^{4-2}_{ANFD-2})} - 1 = 3$.

Example : Let *n*=5 and *ANFD*=3, then $N_{LB} = 2^{(C^{5}_{3} - C^{5-2}_{3-2})} - 1 = 127$.

In the case of quadratic homogenous functions the term $x_1 x_2$ may or may not exist . If $x_1 x_2$ exists then the result will be BB functions and if not the result will be LB functions. The number of BB functions is equal to $2^{C^{n}_{2} - 1}$ and the number of LB functions is equal to $2^{C^{n}_{2} - 1} - 1$.

Example: Let n=4 and $f(x) = a_0x_1x_2 \oplus a_1x_1x_3 \oplus a_2x_1x_4 \oplus a_3x_2x_3 \oplus a_4x_2x_4 \oplus a_5x_3x_4$, then:

(*i*) If $a_0 = 0$, all the combination of $a_i's$ will lead to LB functions with

$$N_{LB} = 2^{C_2^4-1} - 1 = 31.$$

(*ii*) If $a_0 = 1$, then all the combination of $a_i's$ will lead to BB with

$$N_{BB} = 2^{C_2^4-1} = 32.$$

5.5 Average Algebraic Immunity for BB and LB Functions

Algebraic attacks against stream ciphers based on linear feedback shift registers (LFSR's) have been proposed in [54]. Many stream ciphers consist of a linear part, producing a sequence with a large period, usually composed of one or several (LFSR's), and a nonlinear combining function f that produces the output, given the state of the linear part. Algebraic attacks recover the secret key by solving an over defined system of multivariate algebraic equations. These attacks exploit multivariate relations involving key/state bits and output bits of f. If one such relation is found, that is of low degree in the key/state bits, algebraic attacks are very efficient [54].

Low degree relations have been shown to exist for several well known constructions of stream ciphers that are immune to all previously known attacks. Such relations can be derived by multiplying the output function of a stream cipher by a well chosen low degree function, such that the product function is again of low degree. In view of algebraic attacks, low degree multiples of Boolean functions are a basic concern in the design of stream ciphers as well as of block ciphers. In this section we study the average algebraic immunity for BB and LB functions.

5.5.1 Calculating the Algebraic Immunity

In [54], three different scenarios are described under which low degree relation may exist that can be exploit in algebraic attacks. The three scenarios are:

*1)*Assume that there exists a function g of low degree such that the product function is of low degree, i.e. $f * g = h$, where h is a nonzero function of low degree.

2) Assume that there exists a function g of low degree such that $f * g = 0$

*3)*Assume that there exists a function g of high degree such that $f * g = h$, where h is nonzero and of low degree.

In [55] the known scenarios under which low degree multiples exist are reduced and simplified to two scenarios, that are treated differently in algebraic attacks. This simplified description of scenarios leads to precise measure of algebraic immunity of a Boolean function f: The algebraic immunity, *AI(f),* is the minimum value of *ANFD* such that f or $f+1$ admits an annihilating function of degree *ANFD*. The annihilator of f is a non-zero function g such that $f * g = 0$.

The new criteria that f shouldn't have a low algebraic immunity, may be in conflict with some established criteria. This is exemplified for the Maiorana-Mcfarland class [35]. These functions can have high resiliency, high nonlinearity and optimum algebraic degree. Nevertheless it is shown in [55] that such functions can have relatively low algebraic immunity.

In the design of stream cipher we need Boolean functions with high degree annihilator, therefore, it is desirable to have an efficient algorithm for deciding whether a given Boolean function has no low degree annihilator. Such an algorithm is derived in [55] where it allows to successfully decide whether a Boolean function has low degree multiples or not. This represents a significant step towards provable security against algebraic attacks.

Example: Consider the function f=[0111110000000000]. It has the following properties (4,3,4,0). Applying the algorithm proposed in [55], we find that it has an algebraic immunity AI =1 which means that this function has an annihilator of degree 1. From Table 5.6 we see that $f * g = 0$. The annihilator, as shown from Table 5.6, is g = [0000000011111111], which is a linear function.

f	g	$f * g$
0	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
0	0	0
0	0	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0

Table 5.6: The annihilator g of f.

5.5.2 Expected Value of the AI for Randomly Selected Functions

Table 5.7, and Table 5.8 show how the average AI of LB and BB functions are compared to that of a random Boolean function. Exhaustive search is used for $n \leq 5$. For $n > 5$ we tested 5000 LB functions, 5000 BB functions, and 5000 random Boolean functions.

n	3	5	7
AI for BB	1.5	2.2	3.024
AI for LB	1.34375	2.166661	3.02
AI	1.210938	2.044261	3.002

Table 5.7: Average algebraic immunity for odd *n*.

n	2	4	6	8
AI for BB	1	1.875	2.9212	3.997
AI for LB	0.75	1.79052	2.8956	3.998
AI	0.875	1.83847	2.9258	4

Table 5.8: Average algebraic immunity for even *n*.

From our experimental results, we conjecture that the average *AI* is asymptotic to $\frac{n-1}{2}$ for *n* odd (see Figure 5.1), and asymptotic to $\frac{n}{2}$ for *n* even (see Figure 5.2).

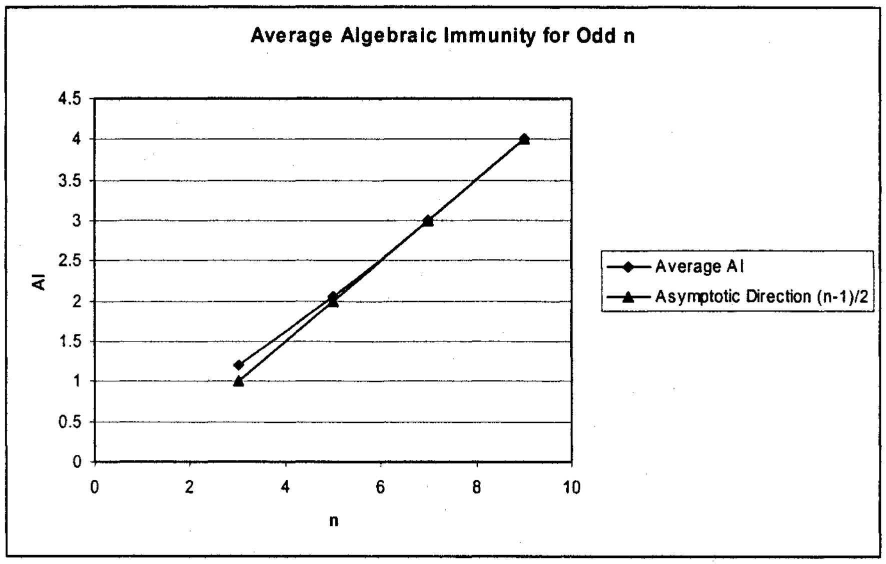

Figure 5.1: Average algebraic immunity for odd *n*.

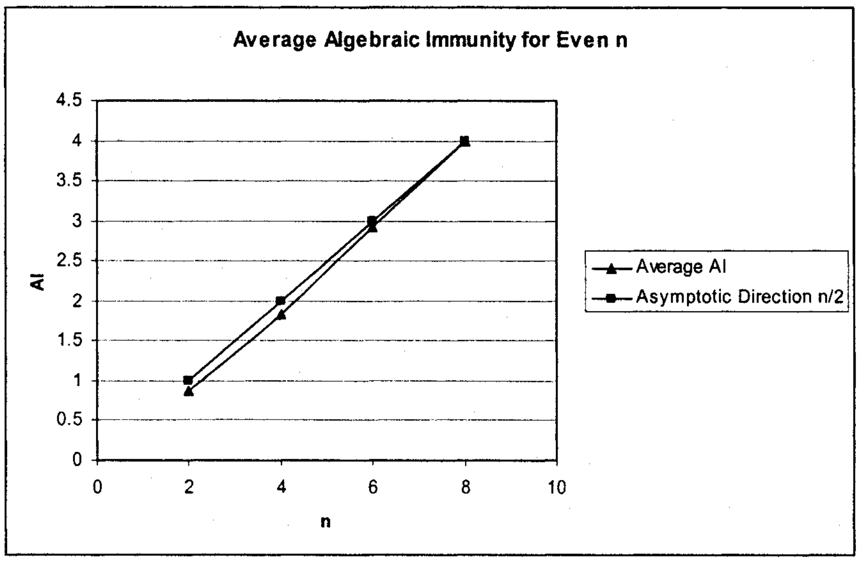

Figure 5.2: Average algebraic immunity for even n.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis we analyzed some criteria that are of great importance in designing cryptographic Boolean functions. In particular,

- We introduced new restrictions on the partial sum of the Walsh transform of binary functions and we extended some of the obtained results to functions defined over *GF(p)*.
- We presented some simple constructions for binary bent functions of length 2^{2k} using a known bent function of length 2^{2k-2}. Our construction techniques are simple and can be performed using hand computations starting with the set of 8 bent functions for $n = 2$.
- We proved that all nonlinear resilient functions with maximum resiliency degree are either bent-based or linear-based. We provided an explicit count for the functions in both classes.

- We introduced new forms for resilient functions over *GF(p)* and we provided a count for these forms.
- We proved that there are no bent-based homogeneous functions with degree >2. We also provided a count for linear-based homogeneous functions.
- We proved that all symmetric bent functions achieving the maximum possible nonlinearity are bent-based.
- Based on our experimental results, we conjecture that the expected value of the Algebraic Immunity for a randomly selected Boolean function is asymptotic to $\frac{n-1}{2}$ for n odd, and asymptotic to $\frac{n}{2}$ for n even.

6.2 Future Work

The results obtained in this thesis can be extended to several directions. For example,

- One can study similar restrictions on the discrete Fourier transform (DFT) of Boolean functions similar to our work on the Walsh transform.
- The restrictions on the partial sum of the Walsh transform of binary functions can be used to speed-up cryptographic Boolean function constructions, based on spectral inversion heuristic search techniques [45].
- Several cryptographic Boolean functions were obtained based on small affine functions [44]. The use of BB functions in similar constructions should be explored.

- All the results obtained in this thesis deal with a single output Boolean function, i.e., $f : Z_2^n \rightarrow Z_2$. Since many cryptographic primitives can be modeled as multi-output functions, it is interesting to extend the results obtained in this thesis to multi-output Boolean functions, i.e., for functions in the form $f : Z_2^n \rightarrow Z_2^m$.

References

[1] D. Kahn, "*The codebreakers: the story of secret writing,*" New York : Macmillan, 1967.

[2] C. E. Shannon, "*Communication theory of secrecy systems,*" Bell Systems Technical Journal, Vol.28, pp. 656–715, 1949.

[3] A. Menezes, P. Oorschot, S. Vanstone, "*Handbook of Applied Cryptography,*" Boca Raton, FL : CRC Press, 1996.

[4] G. J. Simmons, "*Contemporary cryptology, the science of information integrity,*" *IEEE* Press, New York, 1992.

[5] H. Feistel, "*Cryptography and computer privacy,*" Scientific American, Vol. 228, pp. 15–23, 1973.

[6] H. Feistel, W. Notz, and J.L. Smith, "*Some cryptographic techniques for machine-to - machine data communications,*" Proc. of IEEE, Vol. 63, pp. 1545–1554, 1975.

[7] A. M. Youssef, "*Analysis and Design of Block Ciphers,*" Ph.D. thesis, Queen's University, Kingston, Ontario, Canada , December 1997.

[8] M. Matsui, "*Linear cryptanalysis method for DES cipher,*" Advances in Cryptology: Proc. of EUROCRYPT '93, Berlin, pp. 386–397, Springer-Verlag, 1994.

[9] http://www.isrc.qut.edu.au/people/fuller/.

[10] E. Pasalic, T. Johansson, S. Maitra, and P. Sarkar, *"New Constructions of Resilient and Correlation-Immune Boolean Functions achieving Upper Bounds on Nonlinearity,"* WCC2001 International Workshop on Coding and Cryptography, Paris, January 8-12, 2001, Electronic Notes in Discrete Mathematics, Vol. 6, Elsevier Science, 2001.

[11] E. R. Berlekhamp and L. R. Welsh, *"Weight Distribution of the Coset of (32,6) Reed-Muller code,"* IEEE Transactions on Information Theory, IT-18(1), pp. 203-207, January 1972.

[12] J. Maiorana, *"A Classification of the Coset of the Reed-Muller Code R(1,6),"* Mathematics of Computation , Vol. 57, No.195, , pp. 404-414, July 1991.

[13] P. Sarkar and S. Maitra, *"Nonlinearity Bounds and Constructions of Resilient Boolean Functions,"* Proc. of Cypto 2000, pp. 515-532, LNCS 1880, Speinger-Verlag, 2000.

[14] W. Meier and O. Staffelbach, *"Nonlinearity Criteria for Cryptographic Functions,"* In Advances in Cryptology: Proc. of EUROCRYPTt '89, LNCS, Vol. 434, pp. 549–562. Springer-Verlag, 1990.

[15] F. J. MacWilliams and N. J. A. Sloane, *"The Theory of Error Correcting Codes,"* North-Holland Publishing Company, Amsterdam, 1978.

[16] T. Siegenthaler, *"Corrolation Immunity of Nonlinear Combining Functions for Cryptographic Applications,"* IEEE Transactions on Information Theory, Vol. IT-30, pp.776-780, oct. 1984.

[17] M.H. Dawson and S.E. Tavares, "*An expanded set of S-box design criteria based on information theory and its relation to differential attacks,*" Advances in Cryptology: Proc. of EUROCRYPT '91, pp. 352–365, Springer-Verlag, 1992.

[18] J. B. Kam and G. I. Davida, "*Structured design of substitution-permutation encryption networks,*" IEEE Trans. Comp. C-28, pp.747–753, 1979.

[19] R. Forr′e, "*The Strict Avalanche Criterion: Spectral properties of boolean functions and an extended definition,*" Advances in Cryptology: Proc. of CRYPTO '88, pp. 450–468, Springer- Verlag, 1989.

[20] C. M. Adams and S. E. Tavares, "*The use of bent sequences to achieve higher-order Strict Avalanche Criterion in s-box design,*" Technical report, *TR 90–013*, Dept. of Electrical Engineering, Queen's University, Kingston, Ontario, 1990.

[21] B. Preneel, W. V. Leekwijk, L. V. Linden, R. Govaerts, and J. Vandewalle, "*Propagation charachteristic of boolean functions,*" Advances in Cryptology: Proc. of EUROCRYPT '90, pp. 161–173, Springer-Verlag, 1991.

[22] B. Preneel, "*Analysis and Design of Cryptographic Hash Functions,*" PHD thesis , University of Leuven,1994.

[23] O. S . Rothaus, "*On bent functions,*" J. Combinatorial theory, Vol. 20(A), pp. 300-305, 1976.

[24] C. Adams and S. Tavares, "*Generating and Counting Binary Bent Sequences,*" IEEE Transactions on Information Theory, Vol. 36, No. 5, pp.1170-1173, September, 1990.

[25] J. D. Olsen, R. A. Scholtz, and L. R. Welch, *"Bent-function sequences,"* IEEE Transactions on Information Theory, Vol. IT-28 No.6, 1982 .

[26] R. Yarlagadda and J.E. Hershey, *"Analysis and Synthesis of Bent Sequences,"* Computers and Digital Techniques, IEE Proceedings E, Vol. 136 , No. 2 , pp. 112 – 123, March 1989.

[27]C. Carlet, *"A Construction of Bent Functions,"* Finite Fields and Applications, London Mathmatical Society ,Lecture Series 233,Cambridge University Press, pp. 47-58, 1996.

[28] P. V. Kumar, R. A. Scholtz and L. R. Welch, *"Generalized bent functions and their properties,"* J. Combinatorial Theory A 40, pp. 90-107, (1985).

[29] C. Adams, *"Constructing Symmetric Ciphers Using the CAST Design Procedure,"* Designs, Codes and Cryptography, Vol. 12, No. 3, pp. 283-316, November 1997.
[30] X.Zhen,J.Massey, *"A Spectral Characterization of Corrolation-Immune Combining Functions,"* IEEE Transactions on Information Theory, Vol. 34, No. 3, May 1988.

[31] J. Seberry, X. Zhang, *"Construction of bent functions from two known bent functions,"* Journal of combinatorics .9, pp. 21 -35, 1994.

[32] L. Wang , J. Zhang , *"A Best Possible Computable Upper Bound on Bent Functions,"* Journal of uset China.Vol.33, No.2, pp. 113-115, 2004.

[33] Q. Meng, M. Yang, H. Zhang, et.al. *"A novel algorithm enumerating bent functions,"* http://eprint.iacr.org, 2004/274.

[34] J. Fuller, E. Dawson, W. Millan, *"Evolutionary Generation of Bent Functions for Cryptography,"* In Proceedings of Conference on Evolutionary Computation, CEC 2003, Canberra, Australia, 2003.

[35] J. F. Dillon. *"Elementary Hadamard difference sets,"* Proceedings of the Sixth Southeastern Conference on Combinatorics , Graph Theory and Computing, F. Hoffman et al.(Eds), Utilitas Math, pp. 237-249, Winnipeg, Manitoba (1975),.

[36] R. L. McFarland, *"A family of difference sets in non-cyclic groups,"* J. Combinatorial Theory A 15, pp. 1-10, 1973.

[37] Y. Tarannikov, *"New Constructions of Resilient Boolean Functions with Maximal Nonlinearity,"* Fast Software Encryption (FSE), 2001.

[38] P. Camion, C. Carlet, P. Charpin, N. Sendrier, *"On Correlation Immune Functions,"* Advances in Cryptography:Crypto'91, Proceedings, Lecture Notes in Computer Sience,Vol. 576 , pp. 86-100, 1991.

[39] C. Carlet, *"A larger Class of Cryptographic Boolean Functions via a Study of the Maiorana-McFarland Construction,"* Advances in Cryptology - CRYPT0 2002, No. 2442 in Lecture Notes in Computer Science, pp. 549-564, 2002.

[40] C. Carlet, *"On the confusion and diffusion properties of Maiorana-McFarland's and extended Maiorana-McFarland's functions,"* To appear in the Special Issue "Complexity Issues in Coding and Cryptography" of the Journal of Complexity, dedicated to Prof. Harald Niederreiter on the occasion of his 60th birthday.

[41] S. Maitra, P. Sarkar, *"Construction of Nonlieanear Boolean Functions with Important Cryptographic Properties,"* Advances in Cryptology-EUROCRYPT

2000(Lecture Notes in Computer Science), Vol 2551 in Lecture Notes in Computer Science). Berlin, Germany, Vol.1807, pp. 491-512, Springer Verlag, 2000.

[42] E. Pasalic and S. Maitra, *"A Maiorana-McFarland Type Construction for Resilient Boolean Functions on n Variables (n even) with Nonlinearity* > $2^{n-1} - 2^{\frac{n}{2}} + 2^{\frac{n}{2}-2}$*,"* Proceedings of the Workshop on Coding and Cryptography 2003, pp. 365-374, 2003.

[43] E. Pasalic, and S. Maitra, *"Nonlinearity Bounds and Construction of Resilient Boolean Functions,"* Crypto 2000, Berlin. Springler-Verlag. Lecture Notes in Computer Science Vol. 1880, pp. 515-532, 2000.

[44] E. Pasalic, S. Maitra, *"Construction of Nonlinear Resilient Boolean Functions Using "Small" Affine Functions,"* IEEE Transactions on Information Theory, Vol. 50, No. 9, September 2004.

[45] J. Clark, J. Jacob, S. Maitra, and P. Stanica, *"Almost Boolean Functions: The Design of Boolean Functions by Specral Inversion ,"* The 2003 Congress on Evolutionary Computation, Vol. 3, pp. 2173 – 2180, 2003.

[46] S. Maitra, and E. Pasalic, *"Further Construction of Resilient Boolean function with very high nonlinearity,"* IEEE Transactions on Information Theory, Vol.48, pp. 1825-1834, July 2002.

[47] P. Sarkar, *"Spectral Domain of Correlation-Immune and Resilient Boolean Functions ,"* Cryptography ePrint Archive Report, 2000/49, http://eprint.iacr.org.

[48] C.Carlet, *"On the Coset of Weight Divisibility and Nonlinearity of Resilient and Correlation Immune Functions,"* In advances in cryptography CRYPTO 1991, pp. 86-100, Springer Verlag, 1992.

[49] C. Carlet, *"Comment on Generating and Counting Bent Sequences,"* IEEE Transactions on Information Theory, Vol. 40, pp. 600, March 1994.

[50] A. Canteaut and M. Videau, *"Symmetric boolean functions,"* IEEE Transactions on Information Theory, 2005.

[51] I. Wegener, *" The Complexity of Boolean Functions,"* Wiley, 1987.

[52] P. Savicky, *"On the bent functions that are symmetric,"* European Journal of Combinatorics 15 (1994).

[53] S. Maitra, P. Sarkar, *"Maximum nonlinearity of symmetric Boolean functions on odd number of variables,"* IEEE Transactions on Information Theory , Vol. 48, pp. 2626-2630, 2002.

[54] N. Courtois and W. Meier, *"Algebraic Attacks on Stream Ciphers with Linear Feedback,"* Advances in cryptology–EUROCRYPT 2003, Lecture Notes in Computer Science ,Vol. 2656, pp. 346-359, Springer Verlag, 2002.

[55] W.Meier,Enes Pasalic,Cloude Carlet, *"Algebraic Attacks and Decomposition of Boolean Functions,"* EUROCRYPT 2004 , LNCS Vol. 3027,pp. 471-491, 2004.

[56] Z. Saber, A. Youssef, and W. Hamouda, *" On the Construction of Bent Functions, " Proc. of the* IEEE ninth Canadian Workshop on Information Theory, pp. 375-378 , June 2005.