THE DESIGN OF S-BOXES

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Applied Mathematics

---

by

Jennifer Miuling Cheung

Fall 2010

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Jennifer Miuling Cheung:

The Design of S-boxes

J. Carmelo Interlando

J. Carmelo Interlando, Chair
Department of Mathematics and Statistics

Peter Blomgren
Department of Mathematics and Statistics

Carl Eckberg

Carl Eckberg
Department of Computer Science

July 20, 2010

Approval Date

Copyright © 2010
by
Jennifer Miuling Cheung

DEDICATION

To My Children
Mitch and Becca

ABSTRACT OF THE THESIS

The Design of S-boxes
by
Jennifer Miuling Cheung
Master of Science in Applied Mathematics
San Diego State University, 2010

Substitution boxes (aka S-boxes) are the only nonlinear part of a substitution-permutation network as a cryptosystem. Without them, adversaries would compromise the system with ease. Bent functions are a special kind of Boolean functions that achieve maximum nonlinearity. Therefore, it is important to study bent functions since S-Boxes are composed of highly nonlinear Boolean functions. Conventionally, researchers study and analyze Boolean functions in their Algebraic Normal Form. In this work we use cyclotomic cosets to construct nonlinear Boolean functions in their Univariate Polynomial Form. We have three conjectures as our research results and we have found one order 4 bent function with 8 variables. Finally, we analyze the new functions in terms of other design criteria for S-boxes such as strict avalanche and bit independence.We have found a highly nonlinear and balanced Boolean function with 6 variables that fulfills the design criteria and therefore would be a good candidate for constructing an S-box.

TABLE OF CONTENTS

PAGE

ABSTRACT v

LIST OF TABLES vii

LIST OF FIGURES viii

ACKNOWLEDGEMENTS ix

CHAPTER

1 INTRODUCTION AND BACKGROUND ON CRYPTOGRAPHY 1

1.1 What is a Cryptosystem? 2

1.2 Where are S-boxes in a Cryptosystem? 3

1.3 The Data Encryption Standard 4

1.4 Contribution of This Thesis 7

1.5 Overview of This Thesis 8

2 BOOLEAN FUNCTIONS AND S-BOXES 9

2.1 Preliminaries of Boolean Functions 9

2.2 The Nonlinearity of Boolean Functions 12

2.3 Design Criteria for a Good S-Box 14

2.4 Constructing the S-Boxes 15

3 BENT FUNCTIONS 18

3.1 Properties of Bent Functions 18

3.2 Classes of Bent Functions 18

3.3 Constructing Bent Functions from Cyclotomic Cosets 20

3.4 Highly Nonlinear Boolean Functions in Univariate Polynomial Form 21

3.5 Runs Test 23

4 CONCLUSION AND FUTURE WORK 26

BIBLIOGRAPHY 27

APPENDICES

A TRUTH TABLE REPRESENTATION OF S-BOX 1 IN DES CRYPTOSYSTEM 29

B CYCLOTOMIC COSETS 33

LIST OF TABLES

PAGE

Table 1.1 The Navajo Alphabet Code Used During World War II by America's Army as Secret Communication 2

Table 2.1 Evaluating All Possible Combinations of x_1 and x_2 with the Function f 9

Table 2.2 Evaluating All Possible Combinations of x_1, x_2, x_3, and x_4 with The Function f 10

Table 2.3 The Correspondence Between the Primitive Elements and Their 4-bit Input Vectors of the Boolean Function with 4 Varibles and Minterms with an Irreducible Polynomial of $c^4 + c + 1$ 12

Table 2.4 Evaluating All Possible Combinations of x_1 and x_2 with All Linear Functions of f_i 13

Table 2.5 The First S-box from the Date Encryption Standard Cryptosystem with Hexadecimal Entries 14

Table 2.6 The Partial Truth Table of S-box 1 in *DES* System 16

Table 2.7 The Strict Avalanche Criterion of the Four Nonlinear Boolean Functions from the S-box 1 of the DES 17

Table 3.1 The Maximum Degree of Bent Functions in Univariate Polynomial Form for $n = 4, 6, 8, 10$ 20

Table 3.2 The Nonlinearity and Balance of Boolean Functions from Cyclotomic Cosets for $n = 4$ 22

Table 3.3 The Nonlinearity and Balance of Boolean Functions from Cyclotomic Cosets for $n = 6$ 22

Table 3.4 The Strict Avalanche Criterion of C_5, C_3, and C_9 with $n = 6$ if Used to Construct Boolean Functions 23

Table 3.5 The P-values of Boolean Functions from S-box 1 and Cyclotomic Cosets 24

Table A.1 The Truth Table of S-box 1 in DES System 30

LIST OF FIGURES

PAGE

Figure 1.1 An Enigma machine with 3 rotors or scramblers used by the Germans for secret communication. 1

Figure 1.2 An Substitution-Permutation Network with 4 S-boxes and 4 rounds. 5

Figure 1.3 One round of Date Encryption Standard and its function where the 8 S-boxes can be found..... 6

Figure 1.4 Data Encryption Standard algorithm and encryption process. 7

ACKNOWLEDGEMENTS

I would like to thank my thesis committee, Professor Carmelo Interlando, Professor Peter Blomgren, and Professor Carl Eckberg, for their time. Especially Professor Interlando, I am grateful for his patience with me and his guidance.

Special thanks go to Verónica Requena. She has brought her expertise on Bent functions and her connections with other scholars on the subject matter. Our collaboration has advanced my knowledge and skills to analyze such interesting functions. I truly appreciate her time and assistance.

Parts of chapter two in this thesis are from a class project that I had done with Bridget Druken. Therefore I would also like to acknowledge her as well. I remember the countless hours we were trying to learn and compile our LaTeX files to create the presentation output. I really appreciate her contribution, especially her knowledge of the painstaking LaTeX typesetting program.

Towards the end of my research, I was curious about any statistical analysis I could use to test the bit independence of a Boolean function. Thanks to Dr. Barbara Bailey who suggested the runs test, I have provided a statistical perspective on my studies also.

I would also like to show my gratitude to my fellow graduate students for their supports and encouragements along my graduate studies. Their friendships are what I will miss the most and hope to keep forever.

Of course I cannot forget my parents. Without them, I would not have finished my degree. I truly appreciate my mom and dad taking great care of my kids while I am in school.

CHAPTER 1

INTRODUCTION AND BACKGROUND ON CRYPTOGRAPHY

Cryptography is the study of hiding information. In other words, it is the study of writing in secret code or encrypting information that you do not want others to know. From ancient time to modern era, the ability to communicate secretly has been imperative, especially during the time of war. Everyone has heard stories about the German's Enigma machine. Figure 1.1 shows a German Enigma machine with 3 scramblers [1].

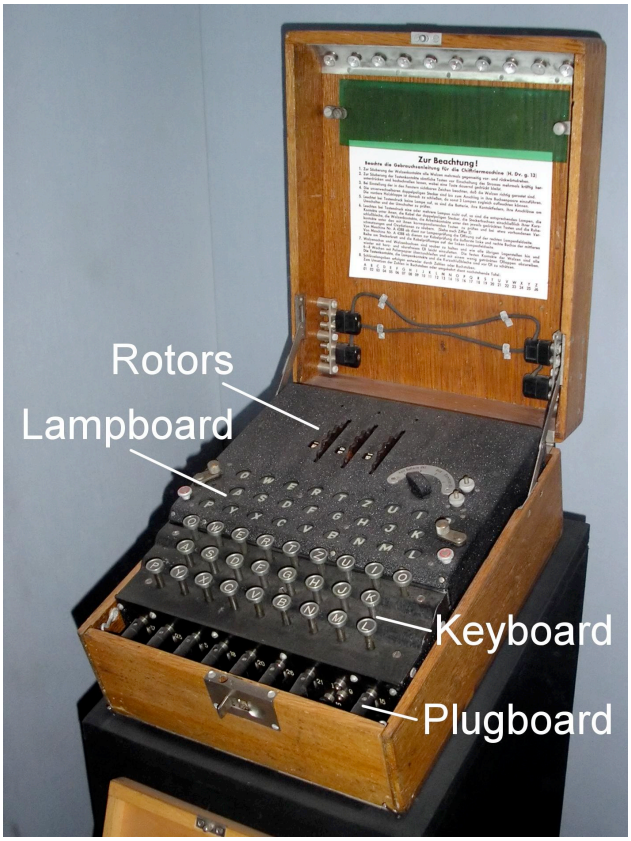


Figure 1.1. An Enigma machine with 3 rotors or scramblers used by the Germans for secret communication.

To crack the code from the Enigma machine, one must know the scrambler orientations, the scrambler arrangements, and which of the six pairs of letters are connected

by the plugboard cables. We had also used the American Indian language, Navajo, as codewords during the World War II. Table 1.1 shows the Navajo alphabet code [2]. These two are good examples of what cryptography is all about. Nevertheless, the mathematics behind cryptography is what made this field even more fascinating.

Table 1.1. The Navajo Alphabet Code Used During World War II by America's Army as Secret Communication

A	Ant	**Wol-la-chee**	**N**	Nut	**Nesh-chee**
B	Bear	**Shush**	**O**	Owl	**Ne-ahs-jsh**
C	Cat	**Moasi**	**P**	Pig	**Bi-sodih**
D	Deer	**Be**	**Q**	Quiver	**Ca-yeilth**
E	Elk	**Dzeh**	**R**	Rabbit	**Gah**
F	Fox	**Ma-e**	**S**	Sheep	**Dibeh**
G	Goat	**Klizzie**	**T**	Turkey	**Than-zie**
H	Horse	**Lin**	**U**	Ute	**No-da-ih**
I	Ice	**Tkin**	**V**	Victor	**A-keh-di-glini**
J	Jackass	**Tkele-cho-gi**	**W**	Weasel	**Gloe-ih**
K	Kid	**Klizzie-yazzi**	**X**	Cross	**Al-an-as-dzoh**
L	Lamb	**Dibeh-yazzi**	**Y**	Yucca	**Tsah-as-zih**
M	Mouse	**Na-as-tso-si**	**Z**	Zinc	**Besh-do-gliz**

1.1 WHAT IS A CRYPTOSYSTEM?

A cryptographic system or a cryptosystem is a system that allows two parties to communicate securely. It contains five elements:

- a finite set of possible plaintexts.
- a finite set of possible ciphertexts.
- a finite set of possible keys.
- a set of encryption functions.
- a set of corresponding decryption functions.

We can represent it mathematically as follows:

Definition 1.1 *Let $\mathcal{P}$ be the plaintext space, $\mathcal{C}$ the ciphertext space, and $\mathcal{K}$ the key space. Let e_k be the encryption function and d_k be the decryption function. Then for each key $k \in \mathcal{K}$, there is an encryption function and a corresponding decryption function such that $d_k(e_k(x)) = x$ for every element $x \in \mathcal{P}$. Each encryption function has to be injective since the decryption must be done unequivocally.*

A cryptosystem can be symmetric (private key cryptography) or asymmetric (public-key cryptography). Symmetric key encryption uses the same key to encrypt and decrypt the texts. Therefore, the key has to be private and the distribution of the key could pose a security problem. Asymmetric key encryption solves this problem by having both a public and a private key for each party in the communication system. Therefore, no secret key exchange is necessary. Unlike the symmetric key encryption, the encryption function and the decryption function in asymmetric key encryption are distinct. While it is relatively easy to use the public key to encrypt messages, it is usually computationally infeasible to decrypt the ciphertexts unless you have the private key.

Block and Stream Ciphers are the two main types of ciphers used in classical cryptography. The difference between the two is the size of the plaintexts processed in each encryption operation. Block ciphers encrypt the plaintexts in blocks of 64 or 128 bits. Stream ciphers encrypt plaintexts one bit at a time. The distinction between these two types of ciphers is not always clear. A stream cipher can be thought of as a block cipher with a very small block size.

1.2 WHERE ARE S-BOXES IN A CRYPTOSYSTEM?

The focus of this thesis is on a type of cipher known as Substitution-Permutation Network (SPN) where S-boxes are utilized to provide the only nonlinear part of the cryptosystem. SPN is a private key cryptosystem. It is a block cipher and will consist of a number of rounds or stages. In a SPN, plaintexts and ciphertexts are both represented by binary vectors of certain length. The two components of a SPN are π_s and π_p. Each permutation π_s is what we call an S-box. It replaces a set of input bits with a different set of bits known as its output bits. We now formally define the SPN cryptosystem:

Definition 1.2 *Let π_s and π_p be the permutation functions such that $\pi_s : \{0,1\}^l \to \{0,1\}^l$ and $\pi_p : \{1, \cdots, lm\} \to \{1, \cdots, lm\}$ where l and m are positive integers, lm is the block length of the cipher, m is the number of S-boxes in the SPN and l is the number of input bits per S-box. Therefore, $\mathcal{P} = \mathcal{C} = \{0,1\}^{lm}$. Let K be the initial key and using the key scheduling algorithm, we can form a key schedule $(K^1, \cdots, K^{Nr+1})$ where Nr is the number of rounds in SPN and $K^i \in \{0,1\}^{lm}$.*

Let $\vec{A} = (x_1, \cdots, x_{lm})$ be the plaintext block of length lm, and let w_r be the state at round r:

$$
\begin{aligned}
w^0 &= \vec{A}, \\
w^0 + K^1 &= u^1, \\
\pi_s(u^1) &= v^1, \\
\pi_p(v^1) &= w^1.
\end{aligned}
$$

Let Nr be the number of rounds of a SPN. The process is repeated for Nr rounds, for any particular round r:

$$
\begin{aligned}
w^{r-1} + K^r &= u^r, \\
\pi_s(u^r) &= v^r, \\
\pi_p(v^r) &= w^r.
\end{aligned}
$$

The ciphertext $\vec{B} = (y_1, \cdots, y_{lm})$ is produced at the last round:

$$
\begin{aligned}
w^{Nr-1} + K^{Nr} &= u^{Nr}, \\
\pi_s(u^{Nr}) &= v^{Nr}, \\
\pi_p(v^{Nr}) &= w^{Nr}, \\
\vec{B} &= w^{Nr} + K^{Nr+1}.
\end{aligned}
$$

Figure 1.2 illustrates the SPN for four rounds with key length of 32 bits and subkey length of 16 bits. It uses four S-boxes. The same S-boxes are used again during each round. Each S-box maps four bits to four bits. It serves us well as an example but in reality, this example would not be secure enough. The key length is small enough to perform an exhaustive key search and therefore break the system. In the following section, we will see an example of a real SPN, namely, the Data Encryption Standard (DES) cryptosystem.

1.3 THE DATA ENCRYPTION STANDARD

The algorithm of the DES cryptosystem was designed and developed by an IBM team during the mid 1970's. In 1977, it was adopted as the national standard by the National Institute of Standard and Technology (NIST). The complete description of DES including the implementation of the system can be found in Federal Information Processing Standards Publication Series 46 [3]. Single DES is no longer considered secure. In FIPS publication 46-3, dated October 25, 1999, Triple Data Encryption Algorithm (TDES) is mentioned to replace single DES. Advanced Encryption Standard (AES) is to coexist with TDES and is believed to provide strong cryptographic security of sensitive information well into the 21st century.

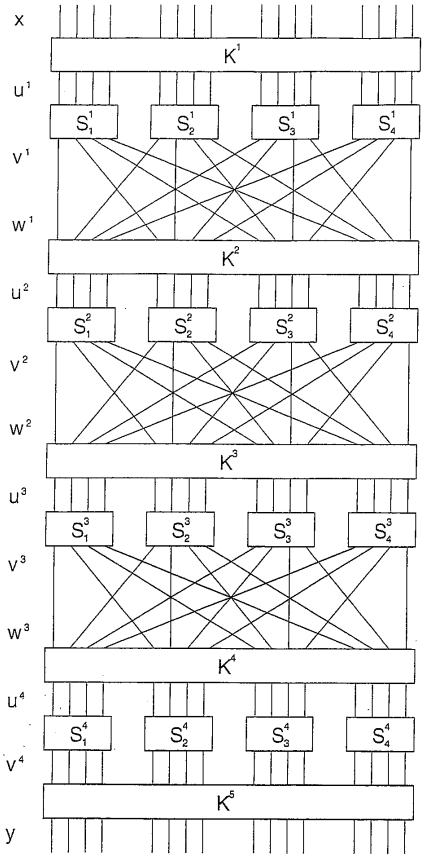
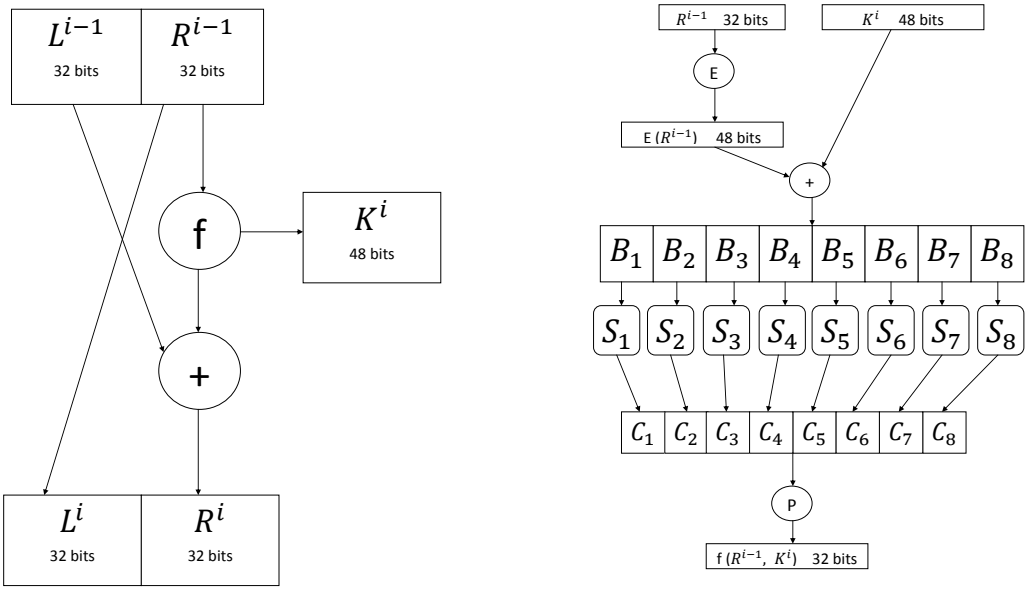


Figure 1.2. An Substitution-Permutation Network with 4 S-boxes and 4 rounds.

DES is a special type of iterated cipher called Feistel cipher. It is a 16-round SPN with a block length of 64 bits. Its key size is 56 bits and 16 round keys of 48 bits are formed from the 56-bit key to be used in DES 16 rounds. There is a fixed initial permutation applied to the plaintext before the first round and an inverse of it is applied after the last round to obtain the ciphertext.

Figure 1.3 shows one round of DES encryption. The plaintext is divided into two halves, 32 bits each. The right half would become the left half of the next round. The right half would expand from 32 bits to 48 bits through an expansion function: $f : \{0,1\}^{32} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32}$ where the nonlinearity of the cryptosystem is located and S-boxes are found. Then it is added to the 48-bit round key. It results in eight 6-bit strings in order to go through the eight S-boxes in the DES. Each S-box takes the 6-bit string and output a 4-bit string, i.e. $\pi_s : \{0,1\}^6 \rightarrow \{0,1\}^4$. The eight 4-bit strings then permuted with the permutation function P and resulting in 32 bits. Now we add the left half of the round to produce the right half of the next round.



(a) 3a. One round of DES.

(b) 3b. The 8 S-boxes within the function.

Figure 1.3. One round of Date Encryption Standard and its function where the 8 S-boxes can be found.

To represent the whole DES system mathematically, let **IP** be the fixed initial permutation and $\mathbf{IP}^{-1}$ be the inverse permutation. Let L and R denote the left and right size of the block cipher, with L^i and R^i be the i state of the DES. K^i represents the i round key used in a round i where $i = 1, \cdots, 16$. Then we can produce each round of cipher block of 64 bits as follow:

$$\begin{aligned} \mathbf{IP}(x) &= L^0 R^0, \\ L^i &= R^{i-1}, \\ R^i &= L^{i-1} \oplus f(R^{i-1}, K^i). \end{aligned}$$

After 16 rounds, we can obtain the ciphertext:

$$y = \mathbf{IP}^{-1}(R^{16} L^{16}).$$

Figure 1.4 summarizes the DES algorithm and shows the encryption process [3]. The Triple DES encrypts each 64-bit block three times with two or three keys using the DES algorithm, therefore increasing the key size from 56 bits to 112 or 168 bits. The Advanced Encryption Standard (AES) is very similar to the SPN that have mentioned before. It has block size of 128 bits, allowable key sizes are 128, 192, and 256 bits. Depends on the key

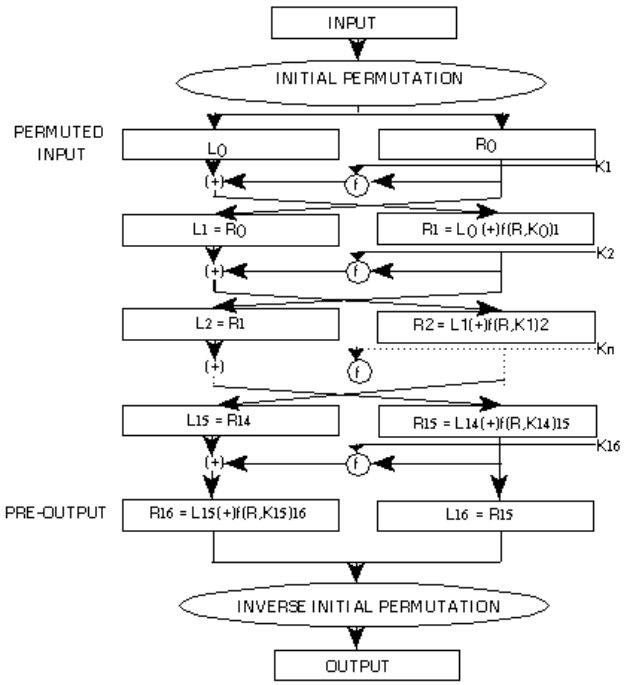


Figure 1.4. Data Encryption Standard algorithm and encryption process.

sizes, its number of round is 10, 12, and 14 respectively. The only difference is it has an additional linear transformation in each round. It has only one S-box, taking a 8 bits input and outputting 8 bits also.

Nonetheless, it is not the focus of this thesis to further discuss these cryptosystems and their algorithms. Now that we know where S-boxes are located and how they are utilized in a cryptosystem, we will see in details how S-boxes work in the next Chapter.

1.4 CONTRIBUTION OF THIS THESIS

As we can see, S-boxes are a very important component to a cryptosystem. In order to design cryptographically good S-boxes, we must study the criteria and the mathematical functions called Boolean functions, behind them. This thesis has provided an overview of these criteria and the study of the Boolean functions in its Univariate Polynomial Form. While many of the criteria have conflicting nature, the most important one is the nonlinearity of the function. During our research, we have found a way to identified good candidate Boolean functions to construct such S-boxes. Nevertheless, we want the output vectors of the functions to demonstrate none statistical pattern. This thesis has also utilized a statistical inference analysis for this matter.

1.5 Overview of This Thesis

Chapter 1 provides necessary background on cryptography and the cryptosystem that contains S-boxes.

In Chapter 2, we will explore more on S-boxes and how they are designed. We will introduce the Boolean functions, their general properties, and how they are responsible for the S-boxes.

In Chapter 3, we present the subset of Boolean functions called bent functions. Since they achieves maximum nonlinearity, much attention has been given to them. Many have studied extensively on bent functions, trying to apply them to the construction of S-boxes. We will present some of our research results here. Our research objective is to construct bent functions in its univariant polynomial form.

Chapter 4 is the conclusion of the thesis. The last chapter will also suggest some of the future work that could be done on the design of S-boxes.

CHAPTER 2

BOOLEAN FUNCTIONS AND S-BOXES

Recall from the previous chapter that the DES S-boxes are represented by a function $f : \{0,1\}^6 \rightarrow \{0,1\}^4$. This function is called a Boolean function. In this chapter, we will discuss and try to understand these functions before we explain further on the design of S-boxes.

2.1 PRELIMINARIES OF BOOLEAN FUNCTIONS

Definition 2.1 *A* Boolean function *is a map $f : \{0,1\}^n \rightarrow \{0,1\}^m$. For simplicity, we will assume $m = 1$. Then the function can be represented as a binary vector $\vec{f}$ of length 2^n where $\vec{f}$ is the rightmost column of the truth table describing the function.*

We denote the set of all Boolean functions as $B_{n,m}$ and B_n when $m = 1$. There exist 2^{2^n} functions in the set of B_n [4]. The truth table is constructed with 2^n rows and n columns. The rows list all possible combinations of n bits. For example, when $n = 2$, we have 4 rows and 2 columns. The rightmost column is the output vector from a Boolean function. We will call this a Boolean vector. Table 2.1 has demostrated the truth table of a Boolean function: $f : \{0,1\}^2 \rightarrow \{0,1\}$.

Table 2.1. Evaluating All Possible Combinations of x_1 and x_2 with the Function f

x_1	x_2	$\vec{f}$
0	0	1
0	1	1
1	0	0
1	1	0

Note that the f is a linear function. Although the truth table is the most common way to express Boolean functions, there are other ways to represent a Boolean function. They are: Algebraic Normal Form (ANF), minterms, and Univariate Polynomial Form (UPF).

In ANF, we represent a Boolean function as a polynomial in $\mathbb{F}_2[x_1, \cdots, x_n]$ with its bitwise sum of its input bits. For example, the ANF representation of Table 2.1 will be

$$f(x_1, x_2) = x_1 \oplus 1.$$

Let us look at another example, say $n = 4$,:

$$f(x_1, x_2, x_3, x_4) = x_1x_2 \oplus x_3x_4.$$

We say that the order (or the degree) of the above function is 2. The order of Boolean functions in ANF is equal to the maximum number of input variables in one single term. When we have the ANF of a Boolean function, we can always construct the truth table to obtain the Boolean vector. Table 2.2 shows the turth table for the function.

Table 2.2. Evaluating All Possible Combinations of x_1, x_2, x_3, and x_4 with The Function f

x_1	x_2	x_3	x_4	$\vec{f}$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

To represent the Boolean function in minterms, let us first understand and define a minterm. Let $a = (a_1, a_2, \cdots, a_n)$, $x = (x_1, x_2, \cdots, x_n)$, where both a and x are $\in \mathbb{Z}_2^n$ [4] [5]. Then the minterm m_a can be expressed as

$$m_a(x) = (1 \oplus a_1 \oplus x_1)(1 \oplus a_2 \oplus x_2) \cdots (1 \oplus a_n \oplus x_n).$$

We also know that $m_a(x) = 1$ if and only if $x = a$. Therefore, we can describe a Boolean function $f(x) = \bigvee_{a \in f^{-1}(1)} m_a(x)$ where $\bigvee$ represents the operation of logical disjunction. Sometimes, it is enough to just specify $f^{-1}(1)$ [4]. Again, once the minterms are known, the truth table can be constructed.

To describe the same Boolean function as in Table 2.1, we would say, the Boolean function has m_0 and m_1. When you have a Boolean function with 4 variables, there are

$2^4 = 16$ possible minterms. From Table 2.2, we can describe the Boolean function with minterms, m_3, m_7, m_{11}, m_{12}, m_{13}, and m_{14}. Each minterm is corresponded with each row in the truth table.

In an UPF, the Boolean function is represented in a finite field, more specifically, the Galois field with 2^n elements. The UPF of a Boolean function is:

$$f(x) = \sum_{i=0}^{2^n-1} a_i x^i,$$

where $a_i \in GF(2^n) = \{0, c, c^2, \cdots, c^{2^n-1}\}$ and $c \in \mathbb{F}_{2^n}$. $\mathbb{F}_{2^n}$ is a primitive element. For example, the Boolean function of 4 variables: $f(x) = x_1x_2 + x_3x_4$ can be expressed as:

$$f(x) = c^3x^{12} + c^5x^{10} + c^6x^9 + c^9x^6 + c^{10}x^5 + c^{12}x^3.$$

The following equation allows us to convert ANF to UPF:

$$f(x) = \sum_{c \in \mathbb{F}_{2^n}} f(c) \times (1 + (x + c)^{2^n-1}).$$

For the above Boolean function with 4 variables, the conversion is done in the $GF(2^4)$. The primitive element c is defined by $c^4 + c + 1$. Then,

$$\begin{array}{lcl} c^4 & \equiv & c + 1, \\ c^5 & \equiv & c^2 + c, \\ c^6 & \equiv & c^3 + c^2, \\ c^7 & \equiv & c^4 + c^3 \equiv c^3 + c + 1, \\ \vdots & \vdots & \vdots \end{array}$$

To see the correspondence between the primitive elements and their 4-bit input vectors and minterms of the Boolean function, we know,

$$\begin{array}{lll} 0 & (0000) & m_0, \\ 1 & (1000) & m_8, \\ c & (0100) & m_4, \\ c^2 & (0010) & m_2, \\ c^3 & (0001) & m_1, \end{array}$$

and the rest of the primitive elements are defined in Table 2.3.

As we mentioned before, the Boolean function has minterms: m_3, m_7, m_{11}, m_{12}, m_{13}, and m_{14}. Accordingly,

$$\begin{array}{lll} f(c^6) = 1, & f(c^{11}) = 1, & f(c^{13}) = 1, \\ f(c^4) = 1, & f(c^7) = 1, & f(c^{10}) = 1. \end{array}$$

Table 2.3. The Correspondence Between the Primitive Elements and Their 4-bit Input Vectors of the Boolean Function with 4 Varibles and Minterms with an Irreducible Polynomial of $c^4 + c + 1$

	c^0	c^1	c^2	c^3	minterm
c^4	1	1	0	0	m_{12}
c^5	0	1	1	0	m_6
c^6	0	0	1	1	m_3
c^7	1	1	0	1	m_{13}
c^8	1	0	1	0	m_{10}
c^9	0	1	0	1	m_5
c^{10}	1	1	1	0	m_{14}
c^{11}	0	1	1	1	m_7
c^{12}	1	1	1	1	m_{15}
c^{13}	1	0	1	1	m_{11}
c^{14}	1	0	0	1	m_9

Now the conversion is almost complete. Using the formula, we have the following function in UPF,

$$\begin{aligned} f(x) &= 1 + (x + c^6)^{15} + 1 + (x + c^{11})^{15} + 1 + (x + c^{13})^{15} \\ &\quad 1 + (x + c^4)^{15} + 1 + (x + c^7)^{15} + 1 + (x + c^{10})^{15}, \end{aligned}$$

and it results in the following UPF after expansion:

$$f(x) = c^3x^{12} + c^5x^{10} + c^6x^9 + c^9x^6 + c^{10}x^5 + c^{12}x^3.$$

We have used this coversion throughout our research.

2.2 THE NONLINEARITY OF BOOLEAN FUNCTIONS

The output of a linear Boolean function can be described as a linear Boolean vector just like the one in Table 2.1. The output of a nonlinear Boolean function can be described as a nonlinear Boolean vector. There are a few more definitions we must know before we discuss how S-boxes are constructed with nonlinear Boolean functions.

Definition 2.2 *The* Hamming weight *(H_w) of a binary vector $\vec{v}$ is the number of 1's in $\vec{v}$.*

Definition 2.3 *The* Hamming distance *(H_d) between two binary vectors of equal length is the number of places for which the corresponding entries are different.*

For example, the Hamming distance between the two binary vectors $x_1 = (1, 1, 0, 0)$ and $x_2 = (1, 0, 1, 0)$ is 2 since x_1 and x_2 differ in the second and third positions.

Notice that the relationship between Hamming weight and Hamming distance is

$$H_d(a, b) = H_w(a \oplus b),$$

where a and b are two binary vectors.

Definition 2.4 *The nonlinearity of a function in the set B_n is defined as the minimum Hamming distance between that function and every linear function in the set.*

In general, the nonlinearity of a function $f \in B_n$ is upper bounded by $2^{n-1} - 2^{\frac{n}{2}-1}$. As an example, let $f \in B_2$ and $f : \{0,1\}^2 \longrightarrow \{0,1\}$. Note that the linear functions of the set are:

$$\begin{aligned} f_1(x_1, x_2) &= 0, & f_2(x_1, x_2) &= 1, \\ f_3(x_1, x_2) &= x_1, & f_4(x_1, x_2) &= x_2, \\ f_5(x_1, x_2) &= x_1 + 1, & f_6(x_1, x_2) &= x_2 + 1, \\ f_7(x_1, x_2) &= x_1 + x_2, & f_8(x_1, x_2) &= x_1 + x_2 + 1. \end{aligned}$$

When we evaluate each of the linear functions, we have the following Table 2.4:

Table 2.4. Evaluating All Possible Combinations of x_1 and x_2 with All Linear Functions of f_i

x_1	x_2	$\vec{f_1}$	$\vec{f_2}$	$\vec{f_3}$	$\vec{f_4}$	$\vec{f_5}$	$\vec{f_6}$	$\vec{f_7}$	$\vec{f_8}$
0	0	0	1	0	0	1	1	0	1
0	1	0	1	0	1	1	0	1	0
1	0	0	1	1	0	0	1	1	0
1	1	0	1	1	1	0	0	0	1

Since $\vec{f_1}, \vec{f_2}, \cdots, \vec{f_8}$ are linear Boolean vectors, and we know there is a total of 2^{2^2} sequences of length 2^2, then there are $16 - 8$ nonlinear Boolean vectors.

We now show an example of finding the Hamming distance of a nonlinear vector in order to find the nonlinearity of the vector. From the above table, we can see that the set of nonlinear vectors g_i = set of all vectors $-$ linear vectors = { (0001), (0010), (0100), (1000), (1110), (1101), (1011), (0111) } [6].

Let $\vec{g_1}$ be a nonlinear vector and $\vec{g_1} = (0001)$, we must take the Hamming distance between $\vec{g_1}$ and each linear vector f_i where $i = (1, 2, \cdots, 8)$. $H_d(\vec{g_1})$ = $\{1, 3, 1, 1, 3, 3, 3, 1\}$. Min($H_d(\vec{g_1})$) = 1. The first element in $H_d(\vec{g_1})$ is found by comparing the vector $\vec{g_1} = (0001)$ to $\vec{f_1} = (0000)$ and noting the number of places where the two vectors differ.

2.3 DESIGN CRITERIA FOR A GOOD S-BOX

What is an S-box then? Table 2.5 shows the first of the eight S-boxes in the DES cryptosystem. One can look at the numbers or entries of the S-boxes and wonder how they are generated. There are attempts to generate those numbers randomly and examine them against the design criteria and guidelines set by the NIST. However, it might result in the construction of weak S-boxes and therefore weaken the cryptosystem. A better and systematic way to generate those entries in the S-boxes is by constructing a nonlinear Boolean function, mapping n input bits to m output bits. A special set of Boolean functions named bent functions can be used to achieve maximum nonlinearity. There are other criteria that must be met in designing the S-boxes. By understanding how to create cryptographically good S-boxes, new S-boxes can be used in the development of new private-key cryptosystems.

Table 2.5. The First S-box from the Date Encryption Standard Cryptosystem with Hexadecimal Entries

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

In the first S-box in the DES system in Table 2.5, we can see that there are 16 columns and the columns are consisted of hexadecimal entries. If we construct a truth table, we will have 6 input columns and 4 output columns of zeros and ones with 2^6 rows. The mapping of the S-box is $f : \{0,1\}^6 \rightarrow \{0,1\}^4$. Therefore, four highly nonlinear balanced Boolean functions compose the S-box. The 6 input bits are split into two groups: the middle four bits indicate the column of the S-box and the two bits on both sides indicate the row of the S-box. We will explain more in details later on how the entries are generated by the four highly nonlinear Boolean functions. But first, let us understand the design criteria of S-boxes.

In general, the following five design criteria must be met [7] [8] for Boolean functions that responsible for a cryptographically good S-box:

1. **Bijection** requires a one-to-one and onto mapping from input vectors to output vectors if the S-box is n by n bit. We will explain later how this criterion is achieved when an S-box is n by m bit instead.
2. **Strict avalanche criterion** occurs if one input bit i is changed, each output bit will change with probability of one half. Strict avalanche requires that if there are any slight changes in the input vector, there will be a significant change in the output vector. To achieve this effect, we will need a function that has a 50% dependency on each of its n input bits.

3. **Bit independence criterion or correlation-immunity** requires that output bits act independently from each others. In other words, there should not be any statistical pattern or statistical dependencies between output bits from the output vectors.
4. **Nonlinearity** requires that the S-box is not a linear mapping from input to output. This would make the cryptosystem susceptible to attacks [9]. If the S-box is constructed with maximally nonlinear Boolean functions, it will give a bad approximation by linear functions thus making a cryptosystem difficult to break.
5. **Balance** means that each Boolean vector responsible for the S-box has the same number of 0's and 1's.

These criteria will meet *most* of the standards set by the National Institute of Standards and Technology. Nevertheless, it is impossible to achieve *all* criteria to their full potential. Their conflicting nature forces us to compromise some of the criteria. For example, correlation immunity conflicts with high nonlinearity and maximum nonlinearity also conflicts with balance [8].

2.4 CONSTRUCTING THE S-BOXES

In general when constructing an S-box, $f : \{0,1\}^n \rightarrow \{0,1\}^m$, with a highly nonlinear function, there are 2^n rows with m columns. A function with its corresponding vector is said to be highly nonlinear when the resulting vector y_i from a function f_i has a high Hamming distance with all the linear vectors in the set of B_n. A truth table is made for the input vector $\vec{x} = (x_1, \cdots, x_n)$. The input vector $\vec{x}$ is evaluated at each Boolean function, f_i where $i = 1, ..., m$. Each Boolean vector $\vec{f_i}$ form the columns of the S-boxes. Therefore, an S-box is comprised of m nonlinear Boolean vectors if the entries of the S-box are binary numbers.

From the earlier example, we see that the nonlinearity of function g_1 is only 1. However, we want the number to be as large as possible. We want to use functions that have a high nonlinearity while still fulfilling all the other criteria at the same time.

Table 2.6 shows a partial truth table representation of the first S-box in the DES cryptosystem. This truth table corresponds to Table 2.5. You can find the complete truth table in Appendix A. Let us look at the first row of the table. When you convert the middle four bits, to decimal, it is 0. When you convert the first and last bits to decimal, it is 0 also. The input bits indicate a row 0 and column 0 entry of the S-box. (Note: All S-boxes start from row 0 and column 0 instead of 1.) The output bits are (1 1 1 0) on the first row of the truth table. Its decimal representation is 14, which is the row 0 column 0 entry of the S-box.

Let's look at another example, the second last row of the truth table has input bits (1 1 1 1 1 0). The middle four bits indicate column 15 and the remaining two bits points out to row 2 of the S-box. The entry of row 2 and column 15 of the S-box is 0, which corresponds to the output bits of (0 0 0 0) on the second last row of the truth table.

Table 2.6. The Partial Truth Table of S-box 1 in *DES* System

x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3	y_4
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	1	0	0
0	0	0	0	1	1	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	1	0	0	0	1	0	1
1	1	1	1	0	1	0	1	1	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	0	1

As mentioned before, y_1, y_2, y_3, and y_4 from the Table 2.6 are nonlinear Boolean vectors. The Hamming weight of each of the four vectors is 32 (see Appendix A). Therefore the four Boolean functions are highly nonlinear and balanced. According to [7], if a Boolean function is highly nonlinear and it has a good avalanche (balanced functions has good avalanche), then the function also fulfills the BIC. Now, let us now look at the SAC.

For SAC, it can be achieved if the Hamming weight of the bitwise sum of the Boolean vector is 16 for $n = 6$ [7]. In general, a function satisfys the SAC when the Hamming weight of the bitwise sum of the Boolean vector $\vec{f}$ of length 2^n is equal to 2^{n-2}. Using the algorithm [7] proposed in their paper, we can calculate the Hamming weight of the bitwise sum of each of the four Boolean vectors of the first S-box in the DES as follow:

Let's $\vec{f}$ be a Boolean vector of 64 bits, & is the bitwise *and* operation, and $\oplus$ is the bitwise *sum* (XOR) operation. All numbers are hexadecimal representation. The symbol $\gg$ represents right shift of the Boolean vector.

1. $H_w\{(\vec{f}$ & 0x00000000FFFFFFFF) $\oplus$ (($\vec{f} \gg 32$) & 0x00000000FFFFFFFF)$\}$ = 16.

2. $H_w\{(\vec{f}$ & 0x0000FFFF0000FFFF) $\oplus$ (($\vec{f} \gg 16$) & 0x0000FFFF0000FFFF)$\}$ = 16.

3. $H_w\{(\vec{f}$ & 0x00FF00FF00FF00FF) $\oplus$ (($\vec{f} \gg 8$) & 0x00FF00FF00FF00FF)$\}$ = 16.

4. $H_w\{(\vec{f}$ & 0x0F0F0F0F0F0F0F0F) $\oplus$ (($\vec{f} \gg 4$) & 0xF0F0F0F0F0F0F0F0)$\}$ = 16.

5. $H_w\{(\vec{f}$ & 0x3333333333333333) $\oplus$ (($\vec{f} \gg 2$) & 0x3333333333333333)$\}$ = 16.

6. $H_w\{(\vec{f}$ & 0x5555555555555555) $\oplus$ (($\vec{f} \gg 1$) & 0x5555555555555555)$\}$ = 16.

If the H_w of each of the six calculations is equal to 16, we say the $\vec{f}$ has fulfilled the SAC. Table 2.7 illustrates the above six calculations of the Boolean functions of the first S-box of the DES.

Table 2.7. The Strict Avalanche Criterion of the Four Nonlinear Boolean Functions from the S-box 1 of the DES

Boolean vectors	$f \gg 32$	$f \gg 16$	$f \gg 8$	$f \gg 4$	$f \gg 2$	$f \gg 1$
$\vec{f_1}$	16	18	15	16	20	20
$\vec{f_2}$	26	18	15	18	17	18
$\vec{f_3}$	18	18	19	14	20	16
$\vec{f_4}$	14	18	13	18	18	17

You might wondering how about the bijection criterion. The mapping from six input bits to four output bits is not bijective for each of the eight S-boxes in the DES. However, if you look at each row of the S-boxes, you can see that the number 0 to 15 only appeares once. It is designed in this particular way to prevent the differential cryptanalysis [9]. Moreover, not all of the eight S-boxes are active in every round of the cryptosystem. The total number of active S-boxes increases as the number of rounds increases [9] [10].

As we mentioned before, all criteria can not be attained to their maximum effect. But [7] mentioned that a balanced function has a fairly good avalanche. Moreover, if a function is highly nonlinear and have a faily good avalanche, the function is also fulfilled the BIC. He also proposes an algorithm to achieve all of the criteria simultaneously by testing each candidate Boolean function against each of the other criteria. The algorithm only deals with n by n S-boxes. The S-boxes they have generated are only fairly good cryptographically. New methods of generating cryptographically good S-boxes are always in demand.

CHAPTER 3

BENT FUNCTIONS

Since nonlinearity is an very important aspect of S-boxes, many have been studies how to generate such S-boxes with highly nonlinear Boolean functions. Some start with a balanced Boolean function and then increase its nonlinearity while trying to maintain other criteria [11]. Others start with a highly nonlinear Boolean function, such as bent function, and decrease its nonlinearity while balancing the bent function [12]. (Bent function is not balanced by nature.) Therefore, construction of highly nonlinear Boolean functions such as bent functions is a research problem. Finding a balanced *and* highly nonlinear Boolean functions is even harder and more valuable for constructing S-boxes. In this chapter, we will learn more about the bent functions and address this reaserch problem.

Definition 3.1 *A bent function is a Boolean function which achieves maximum nonlinearity. Let the bent function be defined as* $f : \{0,1\}^n \rightarrow \{0,1\}^m$ *where* n *is an even number. For simplicity, we will assume* $m = 1$.

3.1 PROPERTIES OF BENT FUNCTIONS

Let $f(x)$ be a bent function of n variables, then the followings are some important properties of bent functions [5]:

- $f(x)$ is not balanced. It consists of 75% ones and 25% zeros or vice verse.
- The Hamming weight of the function is $2^{n-1} \pm 2^{\frac{n}{2}-1}$.
- $1 + f$ is also a bent function of n variables.
- $f(x) + f(a + x)$ is balanced $\forall a \in \mathbb{Z}_2^n$.
- The nonlinearity of the function equales to $2^{n-1} - 2^{\frac{n}{2}-1}$.
- The degree or order (in ANF form) is bounded by $\frac{n}{2}$ for $n \geq 4$.

3.2 CLASSES OF BENT FUNCTIONS

One of the problems of studying bent functions is its vast space. For $n = 2$, there are 8 bent functions. For $n = 4$, there are 896 ent functions. In [5] all of the 896 bent functions have been constructed. Therefore, we know all of the 896 bent functions in their ANF and minterm representations. For $n = 6$, there are 5,425,430,528 bent functions [13]. It is a challenge to enumerate and construct bent functions of higher dimension. Nevertheless, according to [14], bent functions can be grouped into different equivalence classes by affine transformation.

Fuller listed 14 equivalence classes of bent functions with 8 variables and 46 equivalence classes of bent functions with 10 variables. We say that the two functions $g(x)$ and $f(x)$ are belong to the same class if

$$g(\vec{x}) = (f(Ax + b)),$$

where A is all invertible n by n matrices, and x and b are $\in \mathbb{Z}_2^n$

We can further differentiate the bent functions in ANF by their orders as follows [14]:

- for $n = 6$, we have 1 class of order two (quadratic) and 3 classes of order three (cubic);
- for $n = 8$, we have 1 class of order two, 3 classes of order three, and 10 classes of order four;
- for $n = 10$, we have 1 class of order two, 7 classes of order three, 11 classes of order four, and 27 classes of order five.

In fact, we know *all* the homogenous quadratic (order 2) bent functions in ANF for all dimensions. We refer to these quadratic or order two bent functions as Class I and it is in the form of:

$$f(x) = x_1x_2 \oplus x_3x_4 \oplus \cdots \oplus x_{n-1}x_n.$$

Therefore, we can construct *all* bent functions in this form of all dimensions.

When we studied the bent functions of n variables in the UPF, the degree is also bounded by n. Using the equation or transformation formula from ANF to UPF,

$$f(x) = \sum_{c \in \mathbb{F}_{2^n}} f(c) \times (1 + (x + c)^{2^n - 1}).$$

We have converted almost all of the examples of bent functions listed by Fuller for $n = 4, 6, 8, 10$. We have the following conjectures from our research.

Conjecture 3.1 *Let M be the maximum degree of bent functions of n variables in the Univariant Polynomial Form. Then $M = 2^{n-1} + 2^{n-2} + \cdots 2^{n-a}$ for $n \geq 4$, where a is the order of the functions.*

The number of terms in the equation is equaled to the order of the bent function in ANF. For example, a bent function with 8 variables and has order 3 would have three terms. Therefore $M = 2^7 + 2^6 + 2^5 = 224$. Table 3.1 listed the results on maximum degree of bent functions in UPF according to their orders in ANF.

Conjecture 3.2 *Let M_{odd} be the maximum degree of odd ordered bent functions with n variables and M_{even} be the maximum degree of even ordered bent functions with n variables. Then $M_{odd} \equiv 8 \pmod{12}$ and $M_{even} \equiv 0 \pmod{12}$.*

Table 3.1. The Maximum Degree of Bent Functions in Univariate Polynomial Form for $n = 4, 6, 8, 10$

Dimension	Order 2 in ANF	Order 3 in ANF	Order 4 in ANF	Order 5 in ANF
$n = 4$	12	-	-	-
$n = 6$	48	56	-	-
$n = 8$	192	224	240	-
$n = 10$	768	896	960	992

Recall that the UPF of bent functions has coefficient c^i. If we can perform an affine transformation and leaving the equation with the variable x only, it will be useful when we want to investigate the multiplicative complexity of the bent functions in the future. In [8] it was concluded that complexity clearly pays a role in designing cryptographically good S-boxes.

3.3 CONSTRUCTING BENT FUNCTIONS FROM CYCLOTOMIC COSETS

Since all the bent functions of order 2 of all dimensions are known, as well as all the bent functions of 4 variables are known, we are interested in constructing bent functions with higher dimension and higher order. With studying the complexity of these functions in our mind, we are toying with the idea of constructing these functions with cyclotomic cosets.

Definition 3.2 *The operation of multiplying by 2 divides the integers modulus* $2^n - 1$ *into sets called cyclotomic cosets modulus* $2^n - 1$. *The cyclotomic coset containing* i *is* $C_i = \{i \times 2^j \pmod{2^n - 1}; j = 0, 1, 2, \cdots\}$.

There are finitely many elements in each set and there is also a finite number of sets in each dimension. These cyclotomic cosets are easy to construct and the elements in the set are the exponents of the variable x in the function. (All cyclotomic cosets of $n = 4$ and $n = 6$ are listed in Appendix B.) Once we have constructed the UPF of a Boolean function, we tested each function using Property 4 of bent functions mentioned in Section 3.1 to see whether the function is bent or not. Recall $f(x) + f(a + x)$ is balanced $\forall a \in \mathbb{Z}_2^n$. In the case of UPF, $a \in \mathbb{F}_{2^n}$ instead.

We have an interesting result. Using this method, we can also construct all homogenous quadratic bent functions of *all* dimensions. The elements in those cyclotomic cosets responsible for the bent functions share a common trait. Each element has two ones when they are converted to binary numbers. In other words, the Hamming weight of these elements is two when they are represented in their binary forms.

Our result is similar with [15] where the trace functions are involved. We state the following research result:

Conjecture 3.3 *Let $n = 2k$ be the dimension of the homogenous quadratic bent functions. Then $f(x) = \sum_{i=0}^{k-1} x^{2^{(k+1)2^i}}$ is a bent function.*

We have tested up to $n = 12$ by converting from ANF to UPF. It is also consistent with the maximum degree that we have proposed early. The exponents are never larger than M_{even}. Suspecting the strong relationship between the Hamming weight of the elements in their binary forms and the order of the bent functions, we further found that

$$f(x) = x^{15} + x^{30} + x^{60} + x^{120} + x^{135} + x^{195} + x^{225} + x^{240},$$

is an order 4 bent function with 8 variables. All the exponents have a Hamming weight of 4 in their binary forms. You may notice that there are no coefficients c^i in the function. It is because cyclotomic cosets allow one to construct idempotents of degree up to $2^n - 1$. An idempotent $f(x)$ is a polynomial such that $f(x)^2 = f(x)$, therefore $f(x) = 0$ or 1. The coefficients c^i are mapped from $\mathbb{F}_{2^n}$ to 0 or 1.

Unfortunately, this is the only bent function we have found. We tested numerous other possible bent functions and we were unable to found any more. We tested all elements in the cyclotomic cosets up to $n = 10$. We selected a few to test in dimension 12, 14, and 16. However, the number of cyclotomic cosets increases with dimension, and it becomes computationally expensive as the dimension increases.

3.4 HIGHLY NONLINEAR BOOLEAN FUNCTIONS IN UNIVARIATE POLYNOMIAL FORM

In practice, when we construct the S-boxes, we do not want an unbalanced Boolean function. It is because a balanced Boolean function would give a fairly good avalanche and if the function has near-maximally nonlinearity, then it would also fulfill the BIC [7].

Our disappointment quickly turned our focus on how to recognize a Boolean function in UPF has high nonlinearity but not necessary maximum nonlinearity. Recall that the function has to be balanced for all $a \in \mathbb{F}_{2^n}$ if it is a bent function. However, if the function is balanced for most of the a's, not necessary all a, we say the function is highly nonlinear. We also asked such function will be balanced or not. We adjusted the algorithm that we used to test whether a function is bent or not to give us the following information:

1. The number of a's that would result in a balanced function
2. The numbers of 0's and 1's in the function.

Table 3.2 and Table 3.3 list the results for $n = 4$ and $n = 6$. We have tested all cyclotomic cosets in these dimensions. For $n = 4$, the total number of a is 15. The elements

Table 3.2. The Nonlinearity and Balance of Boolean Functions from Cyclotomic Cosets for $n = 4$

$n = 4$				
cyclotomic cosets	# of a's resulted in Bal	# of a's resulted in Non-Bal	# of 0's	# of 1's
C_1	0	15	8	8
C_3	12	3	4	12
C_5 (bent)	15	0	6	10
C_7	9	6	8	8

in C_5 can be used to construct the bent function. For $n = 6$, the total number of a is 63. The elements in C_9 can be used to construct the bent function.

Table 3.3. The Nonlinearity and Balance of Boolean Functions from Cyclotomic Cosets for $n = 6$

$n = 6$				
cyclotomic cosets	# of a's resulted in Bal	# of a's resulted in Non-Bal	# of 0's	# of 1's
C_1	0	63	32	32
C_3	60	3	40	24
C_5	60	3	32	32
C_7	0	63	50	14
C_9(bent)	63	0	28	36
C_{11}	27	36	32	32
C_{13}	27	36	32	32
C_{15}	15	48	40	24
C_{21}	42	21	22	42
C_{23}	0	63	32	32
C_{27}	27	36	28	36
C_{31}	18	45	32	32

As we can see, for $n = 4$, construction of Boolean function using C_3 can produce a highly nonlinear function since 80% of the a's resulted in a balanced function. If we use C_7, we would have a balanced function but its nonlinearity perhaps is not as good as the C_3. C_5 can be used to construct a bent function since all a's produced balanced function and the function itself is not balanced.

For $n = 6$, we can find two highly nonlinear functions if the cosets are used to construct the Boolean functions. If C_5 is used, the Boolean function is also balanced. If C_3 is used, it will produce a non-balanced Boolean function. Once again, we can see all the a's used to test the C_9 produced balanced function. Therefore, it can be used to construct the bent function.

Furthermore, we want to see how these functions fulfill the SAC. Using the algorithm we mentioned above in Section 2.4, we created the truth table after we converted the UPF to ANF. We only created the truth table for C_5, C_3, and C_9 for $n = 6$. The following Table 3.4 shows the SAC result.

Table 3.4. The Strict Avalanche Criterion of C_5, C_3, and C_9 with $n = 6$ if Used to Construct Boolean Functions

cyclotomic cosets	$f \gg 32$	$f \gg 16$	$f \gg 8$	$f \gg 4$	$f \gg 2$	$f \gg 1$
C_5 (bal)	16	16	20	12	16	11
C_3(non-bal)	16	16	16	8	16	15
C_9(bent)	16	16	16	16	16	20

We can see C_9 has the best avalanche. The non-balanced Boolean function has a better avalanche then the balanced one if they are both highly nonlinear. It confirms what we have mentioned earlier. All criteria cannot be achieved fully and balanced function gives only fairly good avalanche. [7] states that these functions should also fulfill the BIC. In the next section, we will introduce a statistical method to test the BIC and verify the claim in [7].

3.5 RUNS TEST

As we mentioned before, the Boolean functions must also fulfill the BIC. We do not want the Boolean vector to give any statistical information or pattern. In other words, we want to make sure that each individual output bit is statistically independent. We also want to make certain that there is no statistical dependencies between two or three or more output bits of the Boolean vector. Runs Test from statistical inference can perform such test. Let us first define Runs Test and the related statistical background.

The Runs Test can be used to test the hypothesis that the elements of the sequence are mutually independent. It is a non-parametric statistical test that examines a randomness hypothesis for a two-valued data sequence by taking the data in the given order. It is perfect to use on our Boolean vectors since they are composed of two values, 0 and 1.

A *run* is defined as a group of successive values of one of the two-values in the data sequence. In our case, runs are groups of successive values of 0 and 1. For example, the vector $\vec{a}$ = (0,0,1,1,1,0,1,0,1,1,0,0) has seven runs; four of which contain 0's and three of them contain 1's.

Let the number of runs in a data sequence be N. There are N_0 runs that contain 0's and N_1 runs contain 1's. Therefore, $N = N_0 + N_1$. If 0's and 1's alternate randomly in the data sequence, then N is a random variable whose conditional distribution, given there is N_0

runs of 0's and N_1 runs of 1's, is approximately normal with [16]

$$\begin{aligned} \mu &= E(N) = \tfrac{2N_0N_1}{N} + 1, \\ \sigma^2 &= Var(N) = \tfrac{(\mu-1)(\mu-2)}{N-1} = \tfrac{2N_0N_1(2N_0N_1-N_0-N_1)}{N^2(N-1)}. \end{aligned}$$

The Null hypothesis, H_0, states that all permutations of N_0 and N_1 have equal probabilities. The alternative hypothesis states otherwise. Therefore, it is a two-sided hypothesis testing. There are two more terms that need to be defined before we present the results of the Runs Test of our Boolean functions.

1. **P-value** is the probability of observing test statistic this extreme or more extreme if the null hypothesis is true. This value measures how much evidence you have against the null hypothesis. Small p-value indicates the outcome measured from the sample data is unlikely to happen if the null hypothesis is true. Small p-value also prompt us to reject the null hypothesis with a predetermine significance level if the p-value $\leq$ significance level.
2. **Significance level** is the decisive p-value we fix in advance. This states when the null hypothesis should be rejected. It is also called type I error; the null hypothesis is true, but we mistakenly reject the null hypothesis. If the p-value is smaller or equal to the signficance level, we reject the null hypothesis.

Runs Test can be performed in R, the statistical software package. It has a default setting of significance level $= 0.05$ which is commonly used in statistical inference. Since the null hypothesis states that all permutations of N_0 and N_1 have equal probabilities, therefore randomly distributed, we do *not* want to reject the null hypothesis. We want large p-value which indicate that we do not have enough evidence to reject the null hypothesis and therefore it stands.

The following Table 3.5 shows the results of the Runs Test on the four Boolean functions which are responsible for the first S-box of DES and our Boolean functions from the cyclotomic cosets with 6 variables.

Table 3.5. The P-values of Boolean Functions from S-box 1 and Cyclotomic Cosets

S-box 1 Boolean functions	P-value	cyclotomic cosets ($n = 6$)	p-value
f_1	0.6143	C_5 (bal)	0.3134
f_2	0.6143	C_3 (non-bal)	2.2 e-16
f_3	0.801	C_9 (bent)	N \ A
f_4	0.2077		

As we can see from the table, the p-values of the four Boolean functions from the first S-box are large and therefore according the the Runs Test, the 0's and 1's alternate randomly

with no particular statistical patterns. The C_5 which can be used to construct a highly nonlinear and balanced Boolean function also has a high p-value. However, we cannot reach the same conclusion for C_3 and C_9. Therefore, it has verified the claim that highly nonlinear and balanced (which give a good avalanche) function implies that the function also fulfills BIC or correlation-immunity and C_5 is a good candidate Boolean function for constructing an S-box.

CHAPTER 4
CONCLUSION AND FUTURE WORK

It is important to study bent functions because of its maximally nonlinearity. However, they are not balance. It is imperative to have a balanced and highly nonlinear Boolean function since balancing a highly nonlinear function is not an easy task. It is even harder to maintain the high nonlinearity while balancing the function. These functions are what we desire for constructing S-boxes.

Using the cyclotomic cosets, we can construct *all* homogenous quadratic (order 2) bent functions of *all* dimensions. We found one order 4 bent function with 8 variables.We found a highly nonlinear and balanced Boolean function of 6 variables using C_5 for construction. It also fulfills other design criteria. We also found another highly nonlinear Boolean function of 6 variables but it is not balance.

The next step of our research would be to gain a better understanding on the hill climb [11] and modified hill climb method [12]. Therefore we can study how to balance a highly nonlinear Boolean function while maintain the high nonlinearity.

Because of the vast space of bent functions in higher dimension, a better computational power or algorithm is needed to test the cyclotomic cosets to see whether we could utilize them to construct bent functions or highly nonlinear Boolean functions. We have tested *all* cyclotomic cosets for $n = 6$ for construction of bent functions. We only selected a few to test for $n = 8, 10$, and 12. More work needs to be done in this area.

Due to the statistical nature of analyzing the S-boxes, further investigation of different statistical approaches besides the Runs Test will be helpful.

BIBLIOGRAPHY

[1] Cybertelecom Federal Internet Law & Policy An Educational Project. Crypto, 2010. `http://www.cybertelecom.org/security/crypto.htm`, accessed March 2010.

[2] S. Singh. *The Code Book*. Anchor Books, New York, USA, 1999.

[3] National Institute of Standards and Technology (NIST). *FIPS PUB 46-3: Data Encryption Standard (DES)*, October 1999.

[4] I. Wegener. *The Complexity of Boolean Functions*. John Wiley & Sons Ltd., Teubner, Stuttgart, 1987.

[5] J. Climent, F. García, and V. Requena. On the construction of bent functions of $n + 2$ variables from bent functions of n variables. *Advances in Mathematics of Communications*, 2(4):421–431, 2008.

[6] C. Adams and S. Tavares. Generating and counting binary bent sequences. *IEEE Transactions on Information Theory*, 36(5):1170–1173, September 1990.

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

[8] J. Cobas and J. Brugos. Complexity-theoretical approaches to the design and analysis of cryptographical boolean functions. In *Computer Aided Systems Theory–EUROCAST 2005*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2005.

[9] D. Coppersmith. The data encryption standard and its strength against attacks. *IBM Journal of Research & Development*, 38(3):243, May 1994.

[10] D. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall/CRC, Ontario, Canada, third edition, 2006.

[11] W. Millan, A. Clark, and Ed Dawson. Boolean function design using hill climbing methods. In *ACISP '99: Proceedings of the 4th Australasian Conference on Information Security and Privacy*, pages 1–11, London, UK, 1999. Springer-Verlag.

[12] Y. Izbenko, V. Kovtun, and A. Kuznetsov. The design of boolean functions by modified hill climbing method. In *TNG '09: Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, pages 356–361, Washington, DC, USA, 2009. IEEE Computer Society.

[13] H. Dobbertin and G. Leander. Cryptographer's toolkit for construction of 8-bit bent functions. Cryptology ePrint Archive, Report 2005/089, 2005. `http://eprint.iacr.org/`.

[14] J. Fuller. *Analysis of Affine Equivalent Boolean Functions for Cryptography*. PhD thesis, Queensland Unversity of Technology, Brisbane, Australia, 2003.

[15] N. Yu and G. Gong. Quadratic bent functions of polynomial forms and their applications to bent sequences. In *23rd Biennial Symposium on Communications*, pages 128–131,

Kigston, Ontario, Canada, 2006.

[16] R. V. Hogg and E. A. Tanis. *Probability and Statistical Inference*. Prentice Hall, Upper Saddle River, New Jersy, sixth edition, 2001.

APPENDIX A

TRUTH TABLE REPRESENTATION OF S-BOX 1 IN DES CRYPTOSYSTEM

TRUTH TABLE REPRESENTATION OF S-BOX 1 IN DES CRYPTOSYSTEM

Table A.1. The Truth Table of S-box 1 in DES System

x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3	y_4
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	1	0	0
0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	1	1	0	1
0	0	0	1	0	1	0	1	1	1
0	0	0	1	1	0	0	0	0	1
0	0	0	1	1	1	0	1	0	0
0	0	1	0	0	0	0	0	1	0
0	0	1	0	0	1	1	1	1	0
0	0	1	0	1	0	1	1	1	1
0	0	1	0	1	1	0	0	1	0
0	0	1	1	0	0	1	0	1	1
0	0	1	1	0	1	1	1	0	1
0	0	1	1	1	0	1	0	0	0
0	0	1	1	1	1	0	0	0	1
0	1	0	0	0	0	0	0	1	1
0	1	0	0	0	1	1	0	1	0
0	1	0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	1	1	0
0	1	0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1	0	0
0	1	0	1	1	0	1	1	0	0
0	1	0	1	1	1	1	0	1	1
0	1	1	0	0	0	0	1	0	1
0	1	1	0	0	1	1	0	0	1
0	1	1	0	1	0	1	0	0	1
0	1	1	0	1	1	0	1	0	1

(table continues)

Table A.1 (continued)

x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3	y_4
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	1	0	0	1	1
0	1	1	1	1	0	0	1	1	1
0	1	1	1	1	1	1	0	0	0
1	0	0	0	0	0	0	1	0	0
1	0	0	0	0	1	1	1	1	1
1	0	0	0	1	0	0	0	0	1
1	0	0	0	1	1	1	1	0	0
1	0	0	1	0	0	1	1	1	0
1	0	0	1	0	1	1	0	0	0
1	0	0	1	1	0	1	0	0	0
1	0	0	1	1	1	0	0	1	0
1	0	1	0	0	0	1	1	0	1
1	0	1	0	0	1	0	1	0	0
1	0	1	0	1	0	0	1	1	0
1	0	1	0	1	1	1	0	0	1
1	0	1	1	0	0	0	0	1	0
1	0	1	1	0	1	0	0	0	1
1	0	1	1	1	0	1	0	1	1
1	0	1	1	1	1	0	1	1	1
1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	1	0	1	0	1
1	1	0	0	1	0	1	1	0	0
1	1	0	0	1	1	1	0	1	1
1	1	0	1	0	0	1	0	0	1
1	1	0	1	0	1	0	0	1	1
1	1	0	1	1	0	0	1	1	1
1	1	0	1	1	1	1	1	1	0
1	1	1	0	0	0	0	0	1	1
1	1	1	0	0	1	1	0	1	0
1	1	1	0	1	0	1	0	1	0
1	1	1	0	1	1	0	0	0	0
1	1	1	1	0	0	0	1	0	1

(table continues)

Table A.1 (continued)

x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3	y_4
1	1	1	1	0	1	0	1	1	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	0	1

APPENDIX B
CYCLOTOMIC COSETS

CYCLOTOMIC COSETS

For $n = 4$:
C_1 = {1,2,4,8}
C_3 = {3,6,12,9}
C_5 = {5,10}
C_7 = {7,14,13,11}

For $n = 6$:
C_1 = {1,2,4,8,16,32}
C_3 = {3,6,12,24,48,33}
C_5 = {5,10,20,40,17,34}
C_7 = {7,14,28,56,49,35}
C_9 = {9,18,36}
C_{11} = {11,22,44,25,50,37}
C_{13} = {13,26,52,41,19,38}
C_{15} = {15,30,60,57,51,39}
C_{21} = {21,42}
C_{23} = {23,46,29,58,53,43}
C_{27} = {27,54,45}
C_{31} = {31,62,61,59,55,47}