



HAL
open science

On the QIC of quadratic APN functions

Shibam Ghosh

► **To cite this version:**

Shibam Ghosh. On the QIC of quadratic APN functions. Discrete Mathematics [cs.DM]. 2020. hal-03135737

HAL Id: hal-03135737

<https://inria.hal.science/hal-03135737v1>

Submitted on 9 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On the QIC of quadratic APN functions

Shibam Ghosh

On the QIC of quadratic APN functions

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Master of Technology in Cryptology and Security

from



Indian Statistical Institute

by

Shibam Ghosh

[Roll No: CrS1801]

under the guidance of

Anne CANTEAUT

Senior Researcher in the Cosmiq project-team

Léo PERRIN

Junior Researcher in the Cosmiq project-team



National Institute for Research in Digital Science and Technology
(Inria Paris)

Institute supervisor

Bimal Kumar Roy

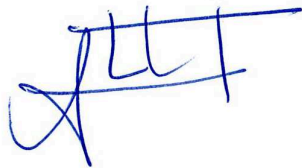
Head of R C Bose Centre for Cryptology and Security

Indian Statistical Institute, Kolkata

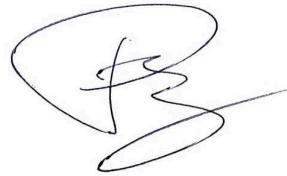
To Anindya

CERTIFICATE

This is to certify that the dissertation entitled “**On the QIC of quadratic APN functions**” submitted by **Shibam Ghosh** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Cryptology and Security** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.



Anne CANTEAUT
Senior Researcher,
National Institute for Research in
Digital Science and Technology (Inria),
Building C, 2nd floor, Room C206
2 rue Simone Iff
CS 42112
75589 Paris Cedex 12, France.
Dated : July, 2020.



Léo PERRIN
Junior Researcher,
National Institute for Research in
Digital Science and Technology (Inria),
Building C, 2nd floor, Room C226
2 rue Simone Iff
CS 42112
75589 Paris Cedex 12, France.
Dated : July, 2020.

Acknowledgments

I would first like to express my profound gratitude to my senior advisor, *Anne CANTEAUT*, Senior Researcher, National Institute for Research in Digital Science and Technology (Inria), Paris, for her patient guidance, enthusiastic encouragement, and useful critiques needed for this research work. Her advice and knowledge helped me in pursuing good research and writing of this thesis. I am privileged to have such a great supervisor as she consistently supported me both academically and personally.

I would also like to thank my advisor, *Léo PERRIN*, Junior Researcher, National Institute for Research in Digital Science and Technology (Inria), Paris, for his valuable advice and assistance in keeping my progress on schedule. I am privileged to have such a friend philosopher and guide in foreign land.

My grateful thanks are also extended to my institute supervisor *Bimal Kumar Roy*, Head of R C Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata, who have paved the way towards my research acumen.

Finally, I thank my parents who introduced me to the first beam of knowledge and supported me in all my endeavours. Last but not the least, I want to thank my friends for keeping me motivated at the toughest period of COVID-19 pandemic situation.

Shibam Ghosh
Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

Vectorial Boolean functions are useful in symmetric cryptography for designing block ciphers among other primitives. One of the main attacks on these ciphers is the differential attack. Differential attacks exploit the highest values in the differential distribution table (DDT). A function is called *Almost Perfect Nonlinear (APN)* if all entries in the DDT belong to $\{0, 2\}$, which is optimal against the differential attack. The search for APN permutations, as well as their classification, has been an open problem for more than 25 years. All these n -variable permutations are known for $n \leq 5$, but the question remains unsolved for large values of n . It has been conjectured for a long time that, when n is even, APN bijective functions do not exist. However, in 2009, Dillon and his coauthors have found an APN permutation of 6 variables.

Our aim on this thesis is to find such functions for larger n . Dillon et al.'s approach was finding an APN permutation from a quadratic APN function which are CCZ-equivalent. Two vectorial Boolean functions are "CCZ-equivalent" if there exists an affine permutation that maps the graph of one function to the other. It preserves the differential properties of a function and thus the APN property. Our approach to find APN permutations will be the same. We propose an idea of representing a quadratic vectorial Boolean function using a cubic structure called *quadratic indicator cube (QIC)*. Also, we describe the criteria related to this cube that is necessary and sufficient for a function to be APN. Then we present some algorithms based on backtracking to change the elements of the cube in such a way that if we start from an APN function it remains APN. We implement our modification algorithm in SageMath and then, in order to get better performances, we also implement it in C++. We also use multithreading in order to get better performances. For $n = 6$ our algorithm outputs 13 EA-equivalence class of functions, which covers all possible classes for $n = 6$.

Keywords: *S-box, Difference distribution table, Almost Perfect Nonlinear function, Big APN problem, CCZ-equivalence, EA-equivalence, Quadratic indicator cube.*

Contents

1	Introduction	6
1.1	Background on Cryptology	6
1.2	Motivation	7
1.3	Attacks on Block Chiphers	8
1.4	Notation and Terminology	10
2	Background	12
2.1	Vectorial Boolean Function	12
2.1.1	Properties of Boolean Functions	12
2.1.2	Walsh Transform	13
2.1.3	Representations of Vectorial Boolean Functions	14
2.1.4	Algebraic Degree of Vectorial Boolean Functions	15
2.2	Differential Cryptanalysis	15
2.2.1	Probability of Differential Propagation	16
2.3	Non-Linearity	20
2.3.1	Bound on Linearity	23
2.4	Vector Spaces Extraction	24
3	CCZ-Equivalence	28
3.1	CCZ-Equivalence and EA-Equivalence	28
3.1.1	EL-mapping and EA-mapping	29
3.1.2	Admissible Affine Permutation	30
3.2	DDT and LAT of CCZ Equivalent Functions	31
3.3	Vector Space of Zeros	33
3.4	TU Projection of a Function	36
4	Quadratic Indicator Cube (QIC)	39
4.1	Notions and Definitions	39
4.1.1	QIC and Derivatives	40
4.1.2	The QIC of APN functions	43
4.1.3	Effect of Composition	45
4.2	Modifying the QIC	47

4.2.1	Modifying a Coordinate	47
4.2.2	Modifying One Sub-diagonal	50
4.2.3	Modify Two Diagonals	53
4.3	Ortho-Derivatives and Ortho-Indicators	59
4.4	Experimental Results	62
4.4.1	Modifying a Coordinate	64
4.4.2	Modifying One Sub-Diagonal	67
4.4.3	Modifying Two Diagonals	68
5	Conclusion	70
A	Examples of 8-bit APN Function	74
A.0.1	Example 1	74
A.0.2	Example 2	74
A.0.3	Example 3	75
A.0.4	Example 4	75
A.0.5	Example 5	75
A.0.6	Example 6	76
B	Machine Configuration	77

List of Figures

2.1	Illustration of difference	16
4.1	The meaning of the different parts of the QIC.	41
4.2	First coordinate(Q_0).	47
4.3	Sub-diagonal.	50
4.4	Upper half of the sub-diagonal	50
4.5	L_1 is the list of all vectors such that $rank(\Delta_{e_1}(F)) = 5$ and $rank(\Delta_{e_4}(F)) = 5$	51
4.6	Two Diagonals	53
4.7	Upper Part	54
4.8	Order of picking vector	54
4.9	Setting vector at position 2.	55
4.10	Setting vector at position 3.	56

List of Tables

4.1	Result of the modification of 2 diagonals. *The Kim mapping belongs to this class	68
4.2	Time required for the modification of 2 diagonals of QIC of size 8. . .	69

Chapter 1

Introduction

Background on Cryptology

Modern *cryptography* involves the study of mathematical techniques for securing digital information, systems, and distributed computations against adversarial attacks [19]. The name cryptography comes from the Greek words “kruptos” (which means hidden) and “graphia” (which means writing). The modern cryptography focuses on the construction of efficient schemes achieving some desired functionality. This functionality can be *privacy*, *integrity*, *authenticity* or any combination of these. On the other hand, *cryptanalysis* is the study of a cryptographic system to find weaknesses in that system. As a whole, both *cryptography* and *cryptanalysis* are included in the subject called *cryptology*. Moreover, we can formally define a cryptosystem as follows [1].

Definition 1 *A cryptosystem is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where the following conditions are satisfied:*

- \mathcal{P} is a finite set of possible plaintexts;
- \mathcal{C} is a finite set of possible ciphertexts;
- \mathcal{K} , the key space, is a finite set of possible keys;
- For each $K \in \mathcal{K}$, there is an encryption rule $e_K \in \mathcal{E}$ and a corresponding decryption rule $d_K \in \mathcal{D}$. Each $e_K : \mathcal{P} \rightarrow \mathcal{C}$ and $d_K : \mathcal{C} \rightarrow \mathcal{P}$ are functions such that $d_K(e_K(x)) = x$ for every plaintext element $x \in \mathcal{P}$.

Many contemporary data encryption principles and concepts were proposed by Claude Elwood Shannon. In his remarkable paper “Communication Theory of Secrecy Systems [27]”, Shannon systematically described the entire cryptographic system and introduced many fundamental concepts.

Cryptosystems are divided into symmetric and asymmetric ones based on the characteristics and usage of the keys. A symmetric cryptosystem uses the same key

for both encryption and decryption while an asymmetric cryptosystem uses different keys for encryption and decryption. There are two kinds of ciphers among symmetric cryptosystems: Stream ciphers and block ciphers. Stream ciphers encrypt the input stream as one character at a time while block ciphers process entire blocks of data in a single operation.

In this thesis, our discussion is mainly related to block ciphers. Block ciphers make it easier to design complicated relationships between the input and the output since any given output character may depend on all input characters. A block cipher E takes as input a key K of a fixed bit-length, and produces a one-to-one map (permutation) from b -bit plaintext to b -bit ciphertext, where b is the bit-length of the fixed block size. Let k be the bit-length of the fixed key size. Then, block ciphers are described as follows:

$$K \in \{0, 1\}^k$$
$$E_K : \{0, 1\}^b \rightarrow \{0, 1\}^b.$$

Since the key size in block ciphers is fixed, it is information theoretically impossible to keep the key secret permanently. Brute force search for the key is always possible. In this attack the attacker tries to recover the k -bit correct key value by simply examining all the 2^k possibilities. But here the main problem is the complexity of the attack. The complexity of an attack can be measured with these three important parameters.

- Time: Time required by an attacker for their computation in offline mode i.e. when not interacting with the oracle.
- Data: Number of queries required by an attacker.
- Memory: Memory required by an attacker.

In the case of a block cipher, these complexities depend on the key size k and the block size b . For modern block ciphers, generic attacks like brute force are impossible due to their high complexity. We will discuss some modern attacks later. But first, we will introduce why Boolean functions are important in cryptography.

Motivation

Modern block ciphers are mainly categorized into two kinds: Feistel structure and substitution-permutation network (SPN) structure. Both designs are a combination of two types of operations: linear operations and nonlinear operations. Linear operations are easy to implement but if all the cipher's operations are linear, the resulting cipher is also linear and then an attacker can find linear equations among plaintexts, ciphertexts and keys. This makes a cipher 'insecure' because solving this equation can reveal the key. So we need to introduce non-linear operations.

In his paper [26], Shannon introduced a basic paradigm called the *confusion-diffusion* paradigm. The basic idea is to construct a block cipher E with large block length from many smaller ‘random-looking’ permutations $\{e_i\}$ with small block length. Let E be a block cipher with block length of $b = mn$ bits. Then the key K for E will specify n permutations e_1, e_2, \dots, e_n , each have an m -bit block length. So if $x \in \{0, 1\}^b$, then we have

$$E_K(x) = e_1(x_1) \parallel, \dots, \parallel e_n(x_n),$$

where each x_i is m -bit. These round functions $\{e_i\}$ are said to introduce *confusion* into E . After this the output bits of E are permuted using a mixing permutation. This step is called *diffusion*. The combination of confusion and diffusion is called a *round*.

So the strategy of the block cipher design is alternately applying the nonlinear and linear operations. Substitution box, or S-box is an important component for non-linearity, which produces *confusion*. The S-box, is a pre-specified mapping from the input values to the output values. When the cipher is designed for more resource-constrained hardware such as micro-controllers and thus the implementation cost is a big issue, an S-box is a good option as a non-linear component. Boolean and vectorial Boolean functions play an essential role in designing S-boxes. According to Shannon’s principles of confusion and diffusion [26], these functions should be designed in such a way that there will be no ‘mutual information’ between the input and output of the cipher. Different characteristics of a function can help us to achieve this and lack of those characteristics leads to different types of cryptographic attacks. Also, because of the fact that the values of these different characteristics mutually restrict one another, it is in general impossible to find functions which are optimal with respect to all of these characteristics. This motivates the study of different classes of optimal Boolean functions along with their construction.

Attacks on Block Chiphers

The power of an attacker depends on the attack model in which they try to attack. These are some example of possible attack models [25].

- Ciphertext only attack: The attacker has the power to observe only ciphertexts produced by the encryption system.
- Known plaintext attack: The attacker has the power to observe pairs of plaintext and the corresponding ciphertext.
- Chosen plaintext attack: The attacker has an access to the encryption system and can encrypt any plaintext of the attacker’s choice.

- Chosen ciphertext attack: The attacker has an access to the decryption system and can decrypt any ciphertext of the attacker’s choice.

In this section we will give a general overview of an important attack on block ciphers.

Differential Attack

Differential attack was introduced by Biham and Shamir [3]. This attack considers difference between two plaintexts. This assumes the existence of an ordered pair (a, b) such that the condition $F(x \oplus a) \oplus F(x) = b$ holds with high probability, where F is a function used as an S-box. Attacker defines a table, called *differential distribution table (DDT)*, which is a matrix with element at (a, b) position defined by:

$$\delta_F(a, b) = \#\{x \in \mathbb{F}_2^n : F(x \oplus a) \oplus F(x) = b\}.$$

The related criterion on a function F used as an S-box is that the values in its DDT must be as uniformly distributed as possible for all non-zero a , because higher values of $\delta_F(a, b)$ help the attacker. This gives the main motivation of our work. The maximum value in the DDT for $a \neq 0$ is called the *differential uniformity* [23]. A vectorial Boolean function is called *almost perfect nonlinear (APN)* [24] if all the values in the position (a, b) for all $a \neq 0$ of the DDT of F are at most 2 i.e the differential uniformity is 2. The search for APN permutations, as well as their classification, has been an open problem for more than 25 years. All these permutations are known for $n \leq 5$ [5], but the question remains unsolved for large values of n . It has been conjectured for a long time that, when n is even, APN bijective functions do not exist. However, in 2009, Dillon and his coauthors have found an APN permutation of 6 variables. [6]. But when n is odd, we have known example of APN permutation (see Section 2.2).

Our Contribution

Our aim in this thesis is finding infinite families of APN functions. Dillon et al.’s approach [6] was finding an APN permutation from a quadratic APN function. Our approach to study and find APN permutations will be the same. For these we need to find APN functions. In this work we describe a way of deriving new APN functions from a known one. We introduce a method to represent a quadratic vectorial Boolean function with a cubic structure called *quadratic indicator cube (QIC)*. We present some algorithms to change the elements of the cube in such a way that if we start from an APN function it remains APN. With this approach we will try to find new APN functions which are in different “CCZ-equivalence” class. Two vectorial Boolean functions are “CCZ-equivalent” if there exists an affine permutation that maps the graph of one function to the other. This equivalence preserves the differential properties of a function. So the final target will be finding an APN function which is “CCZ-equivalent” to a permutation. In that case the permutation will be an

APN permutation. We provide three ways to modify the QIC. We have primarily implemented our algorithms in SageMath and then for better performance we have also implemented in C++. For 6-variable APN function our algorithms have found many new functions but For 8-bit APN functions our algorithm is running successfully, but so far we have not found any new function.

Outline of this thesis

Chapter 2 contains an overview of vectorial Boolean functions and their properties. Also in this chapter we define the ‘Big APN problem’ in detail. In Chapter 3 we discuss “CCZ-equivalence” and how to find a CCZ-equivalent permutation from a function. Chapter 4 contains our main contribution: The construction of the QIC, the APN criteria related to the QIC, some modification procedures of the QIC and our experimental results. Chapter 5 concludes the thesis.

Notation and Terminology

- When the dimension of a vector space is clear from context we use e_i to denote the i th standard basis vector (the vector with a single 1 in position i and zeros everywhere else).
- $0 \in \mathbb{F}_2^d$ is to denote all-zeros vector.
- If $a, b \in \mathbb{F}_2$ then $a + b = (a + b) \pmod{2}$. Sometimes we use \oplus for this operation.
- If $V = (v_0, v_1, \dots, v_{d-1})$ and $W = (w_0, w_1, \dots, w_{d-1})$ are two vectors in \mathbb{F}_2^d then

$$V \cdot W = v_0w_0 + \dots + v_{d-1}w_{d-1}$$

is the dot product of V and W .

- A^T denotes the transpose of a matrix A .
- If $S = \{V_0, V_1, \dots, V_{k-1}\}$ is a set of vectors then $\text{span}\{S\} = \text{span}\{V_0, V_1, \dots, V_{k-1}\}$ denotes the vector space spanned by S .
- The characteristic function of the function $F : \mathbb{F}_2^d \rightarrow \mathbb{F}_2^d$ is denoted by θ_F is the function from $\mathbb{F}_2^d \times \mathbb{F}_2^d$ to \mathbb{F}_2 defined by

$$\theta_F(x, y) \mapsto \begin{cases} 1, & \text{if } y = F(x) \\ 0, & \text{otherwise.} \end{cases}$$

- The Hadamard-Walsh transform or discrete Fourier transform of the function $f : \mathbb{F}_2^d \rightarrow \mathbb{R}$ is denoted by \hat{f} and is the function from $\mathbb{F}_2^d \times \mathbb{F}_2^d$ to \mathbb{F}_2 defined by

$$\hat{f}(w) = \sum_{x \in \mathbb{F}_2^d} f(x)(-1)^{x \cdot w}.$$

- If f and g two function over \mathbb{F}_2^d , we denote the convolution product of f and g as $f \otimes g$, defined by

$$(f \otimes g)(a) = \sum_{x \in \mathbb{F}_2^d} f(x)g(a + x).$$

- For any $u = (u_0, \dots, u_{n-1}) \in \mathbb{F}_2^n$, x^u denotes the monomial in $\mathbb{F}_2[x_0, x_1, \dots, x_{n-1}]/(x_0^2 + x_0, x_1^2 + x_1, \dots, x_{n-1}^2 + x_{n-1})$ defined by

$$\prod_{i=0}^{n-1} x_i^{u_i}.$$

Chapter 2

Background

This chapter starts with an overview of Boolean functions in Section 2.1. In Section 2.2 we describe about differential cryptanalysis and some characterizations of Almost Perfect Non-linear functions. In Section 2.3 we present a bound on linearity of Boolean functions. We finish this chapter with a discussion on finding vector spaces of specified dimension that are completely contained in some set in Section 2.4. We will use this process of finding vector space in Section 3.4 to find permutation from a function, which are CCZ-equivalent.

Vectorial Boolean Function

Vectorial Boolean functions are useful in symmetric cryptography for designing block ciphers. For example “S-boxes” in block ciphers are vectorial Boolean functions. Here we will provide the definition and some properties of vectorial Boolean functions.

Definition 2 *A Vectorial Boolean function is of the form*

$$\begin{cases} F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n \\ x = (x_0, x_1, \dots, x_{n-1}) \mapsto (F_0(x), F_1(x), \dots, F_{n-1}(x)). \end{cases}$$

Here each F_i is a Boolean function from \mathbb{F}_2^n to \mathbb{F}_2

As we can identify a finite field \mathbb{F}_{2^n} with the vector space \mathbb{F}_2^n , then we can define vectorial Boolean function over \mathbb{F}_{2^n} . In this case any input $x \in \mathbb{F}_{2^n}$ will be considered as a vector $(x_0, x_1, \dots, x_{n-1})$. In this thesis, vectorial Boolean function means it is defined over \mathbb{F}_2^n to itself unless stated otherwise. Similarly Boolean function means it is defined from \mathbb{F}_2^n to \mathbb{F}_2 .

Properties of Boolean Functions

Definition 3 *If $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and $F(x) = (F_0(x), F_1(x), \dots, F_{n-1}(x))$ then each F_i is called a coordinate. Also, F has $2^n - 1$ components which are of the form $x \mapsto a \cdot F(x)$,*

for each nonzero a in \mathbb{F}_2^n .

We will denote vectorial Boolean functions with uppercase letters F, G etc. We will denote Boolean functions with lowercase letters f, g etc. Also the i th coordinate of F will be denoted as F_i and the component $x \mapsto a \cdot F(x)$ will be denoted as F_a .

Definition 4 *The support of a Boolean function f is denoted by $\text{supp}(f)$, the set*

$$\text{supp}(f) = \{x \in \mathbb{F}_2^n : f(x) = 1\}.$$

The size of $\text{supp}(f)$ is the Hamming weight of f and is denoted by $\text{wt}(f)$.

Definition 5 *If $\text{wt}(f) = 2^{n-1}$ for an n -variable Boolean function then f is called a balanced function. So a balanced function f takes the output values 0 and 1 the same number of times.*

Also we can define the distance between two Boolean functions f and g as

$$d(f, g) = \#\{x \in \mathbb{F}_2^n : f(x) \neq g(x)\} = \text{wt}(f + g).$$

Walsh Transform

Here we will discuss an important transformation, called *discrete Fourier transform*. Suppose f is a Boolean function then we can define another integer function related to f called the *sign function*, denoted by f_χ ,

$$f_\chi(x) = (-1)^{f(x)}.$$

Definition 6 *The Walsh transform of a Boolean function f , denoted by W_f , is a Fourier transform of its sign function,*

$$W_f(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) + a \cdot x}, \quad a \in \mathbb{F}_2^n.$$

Similarly, for a vectorial Boolean function F we can define the *Walsh transform* as

$$W_F(a, b) = \sum_{x \in \mathbb{F}_2^n} (-1)^{b \cdot F(x) + a \cdot x}, \quad a, b \in \mathbb{F}_2^n.$$

We can also define the *Walsh transform* of a function on finite field \mathbb{F}_{2^n} . For this, we need a function called the *trace function*.

Definition 7 *The function $\text{tr} : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_2$ is the trace function defined by*

$$\text{tr}(x) = x + x^2 + x^{2^2} + \cdots + x^{2^{n-1}}.$$

Suppose $f : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_2$, then the *Walsh transform* of f is

$$W_f(a) = \sum_{x \in \mathbb{F}_{2^n}} (-1)^{f(x) + \text{tr}(ax)}, \quad a \in \mathbb{F}_{2^n}.$$

Also if $F : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$, then the *Walsh transform* of F is

$$W_F(a, b) = \sum_{x \in \mathbb{F}_{2^n}} (-1)^{\text{tr}(bF(x)) + \text{tr}(ax)}, \quad a, b \in \mathbb{F}_{2^n}.$$

Definition 8 *The Walsh spectrum is the set*

$$\text{WS}_F = \{ W_F(a, b) : a, b \in \mathbb{F}_2^n \},$$

and the extended Walsh spectrum is the set

$$\text{EWS}_F = \{ | W_F(a, b) | : a, b \in \mathbb{F}_2^n \}.$$

Proposition 1 Parseval's equality [8]: *If $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ is a Boolean function then $\sum_{a \in \mathbb{F}_2^n} W_f(a)^2 = 2^{2n}$.*

Proof:

From the definition of the Walsh transform of f , we have:

$$\begin{aligned} \sum_{a \in \mathbb{F}_2^n} W_f(a)^2 &= \sum_{x, y \in \mathbb{F}_2^n} (-1)^{f(x) + f(y)} \sum_{a \in \mathbb{F}_2^n} (-1)^{a \cdot (x+y)} \\ &= \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) + f(x)} 2^n \\ &= 2^n \cdot 2^n \\ &= 2^{2n}. \end{aligned}$$

□

Representations of Vectorial Boolean Functions

A vectorial Boolean function can be represented by each of its coordinates which are functions from \mathbb{F}_2^n to \mathbb{F}_2 . One important representation of a Boolean function is its algebraic normal form. In this case a Boolean function will be represented as a multivariate polynomial. Note that any component f takes its values in \mathbb{F}_2 and if $a \in \mathbb{F}_2$ then $a^2 + a = 0$. So the values of f must be considered modulo $a^2 + a$.

If f is a Boolean function then there exists a unique multivariate polynomial in $\mathbb{F}_2[x_0, x_1, \dots, x_{n-1}] / (x_0^2 + x_0, x_1^2 + x_1, \dots, x_{n-1}^2 + x_{n-1})$ such that

$$f(x_0, x_1, \dots, x_{n-1}) = \sum_{v \in \mathbb{F}_2^n} a_v x^v, \quad \text{with } a_v \in \mathbb{F}_2.$$

This multivariate polynomial is called the algebraic normal form (ANF) of f . For example let us consider the function $F(x) = x^3$ from \mathbb{F}_{2^6} to itself. Now we can take this as a function from \mathbb{F}_2^6 to itself with input variables (x_0, x_1, \dots, x_5) . Let us consider the irreducible polynomial corresponding to the field \mathbb{F}_{2^6} is $x^6 + x^4 + x^3 + x + 1$. Then the algebraic normal form of its 6 coordinates is as follows

- $F_0(x) = x_0x_3 + x_0x_4 + x_0x_5 + x_0 + x_1x_5 + x_2x_3 + x_2 + x_3 + x_4x_5 + x_4$
- $F_1(x) = x_0x_1 + x_0x_3 + x_0x_4 + x_1x_3 + x_1x_4 + x_1x_5 + x_2x_4 + x_2 + x_4x_5$
- $F_2(x) = x_0x_1 + x_0x_2 + x_0x_4 + x_1x_3 + x_1x_4 + x_1x_5 + x_2x_4 + x_2x_5 + x_3 + x_5$
- $F_3(x) = x_1x_4 + x_1x_5 + x_1 + x_2x_5 + x_2 + x_3x_4 + x_4x_5 + x_4$
- $F_4(x) = x_0x_2 + x_0x_3 + x_0x_5 + x_1x_2 + x_1x_3 + x_2x_5 + x_2 + x_3x_5 + x_3 + x_4x_5$
- $F_5(x) = x_0x_4 + x_1x_2 + x_1x_4 + x_3x_5 + x_3 + x_4 + x_5$.

The following result is important to compute the ANF :

Theorem 1 *If $f(x_0, x_1, \dots, x_{n-1}) = \sum_{v \in \mathbb{F}_2^n} a_v x^v$ with $a_v \in \mathbb{F}_2$, then $a_v = \sum_{x \preceq v} f(x)$ and $f(v) = \sum_{x \preceq v} a_x$ where $x \preceq y$ if and only if $x_i \leq y_i$ for all $0 \leq i \leq n - 1$.*

There is an efficient algorithm to compute the ANF described in [8].

Algebraic Degree of Vectorial Boolean Functions

Definition 9 *The algebraic degree of a Boolean function is the maximum number of variables in a term of its ANF. For a vectorial Boolean function the algebraic degree is the maximum algebraic degree of its coordinates.*

So for the above function each coordinate has algebraic degree 2 and then the algebraic degree of the function is also 2. In this thesis we will mainly focus on quadratic (degree 2) functions.

Differential Cryptanalysis

According to Shannon's principles of confusion and diffusion [26], an S-box should have the property that the relation between the input and output of the cipher should be complicated. One important attack using this type of relation is differential cryptanalysis. Differential cryptanalysis [3] was established by Biham and Shamir.

Differential cryptanalysis focuses on differences between two inputs rather than individual values. The reason behind this is that in many deterministic computations differential propagation can be predicted even without knowing the key value. This

fact suggests a method of cryptanalysis [17]. In many block ciphers, the key or the values derived from the key is XORed to the plain text. So we can get the following property:

$$\begin{aligned}
 \Delta x &= x \oplus x' \\
 &= x \oplus k \oplus k \oplus x' \\
 &= y \oplus y' \\
 &= \Delta y.
 \end{aligned}
 \tag{2.1}$$

Although each of the values of $x \oplus k$ and $x' \oplus k$ is unknown, the attacker still knows their difference irrespective of the key value k .

Suppose that two input messages x and x' are processed by the same function F . Namely, $F(x)$ and $F(x')$ are computed. The differential cryptanalysis focuses on the difference of two computations i.e. $F(x) \oplus F(x')$. The following figure will illustrate the concept:

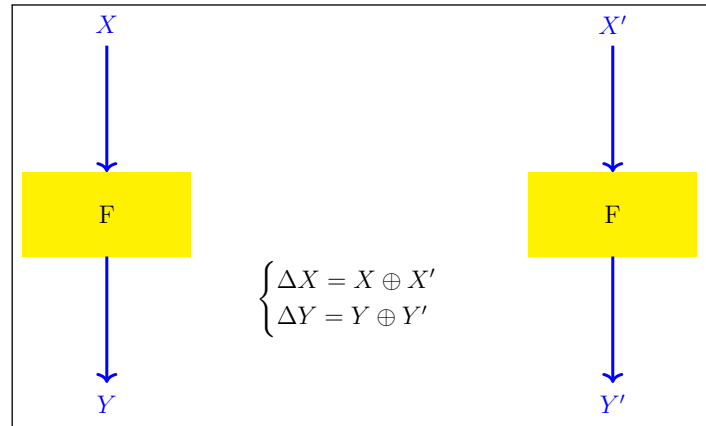


Figure 2.1: Illustration of difference

The fact that the difference Δx propagated to Δy (denoted by $\Delta x \rightarrow \Delta y$) after an operation is called a *differential*. We are interested in the probability of this event i.e. $Pr(\Delta x \rightarrow \Delta y)$.

Probability of Differential Propagation

The attacker usually cannot compute the difference of $F(x)$ and $F(x')$ without knowing the key K . That's why we are interested in $Pr(\Delta x \rightarrow \Delta y)$ for some input difference Δx and output difference Δy rather than for the specific input x and output y . Also, we can see that if the function F is an affine function then $Pr(\Delta x \rightarrow \Delta y) = 1$ or 0. But for non-linear computation this property does not hold. Observing the

probability of differential propagation is the main task for differential cryptanalysis [28].

Also we can compute the probability as:

$$Pr(\Delta x \rightarrow \Delta y) = \frac{\#\{x : F(x) \oplus F(x \oplus \Delta x) = \Delta y\}}{2^b},$$

where $x \in \mathbb{F}_2^b$.

To represent these probabilities we can use a matrix called Difference Distribution table (DDT).

Definition 10 *The Difference Distribution Table (DDT) is a $2^n \times 2^n$ matrix with elements at (α, β) position defined by:*

$$\delta_F(\alpha, \beta) = \#\{x \in \mathbb{F}_2^n : F(x \oplus \alpha) \oplus F(x) = \beta\}$$

Obviously higher values in DDT help the attacker. This idea motivates the following definition, introduced by Kaisa Nyberg in [22].

Definition 11 (Differential uniformity) *It is the maximum value in the DDT when $\alpha \neq 0$ and is denoted by δ_F for a function F . So*

$$\delta_F = \max_{\alpha \neq 0} \delta_F(\alpha, \beta).$$

Thus in view of differential cryptanalysis we are getting an important cryptographic property of vectorial Boolean function that is its differential uniformity should be low to prevent differential attack. Clearly, the value of δ_F for any non-zero function F is at least 2 because all values in the DDT are even.

Definition 12 [24] *A function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is almost perfect nonlinear (APN) if and only if its differential uniformity is 2 i.e for all $a \in \mathbb{F}_2^n \setminus \{0\}$ and $b \in \mathbb{F}_2^n$,*

$$\#\{x \in \mathbb{F}_2^n : F(x + a) + F(x) = b\} \leq 2.$$

We will discuss more about APN functions and the criteria to be an APN function in Chapter 4. Here we will present some characterizations of APN functions. Most of this part is from [2]. Here let us first introduce the following notation as in [2] for a Boolean function f :

$$\mathcal{F}(f) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)}.$$

So obviously $W_f(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) + a \cdot x} = \mathcal{F}(f + \phi_a)$, where $\phi_a(x) = a \cdot x$. Also we get $\mathcal{F}(f) = W_f(0)$. So

$$\begin{aligned} \mathcal{F}(f) &= \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} \\ &= \#\{x : f(x) = 0\} - \#\{x : f(x) = 1\} \\ &= 2^n - wt(f). \end{aligned}$$

Definition 13 *The derivative of $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ with respect to any element $a \in \mathbb{F}_2^n$ at $x \in \mathbb{F}_2^n$ is $\Delta_a F(x) = F(x+a) + F(x)$.*

Theorem 2 *(Theorem 2 of [2]) Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and $F_\lambda, \lambda \in \mathbb{F}_2^n$ denote its components. Then, for any nonzero $a \in \mathbb{F}_2^n$*

$$\sum_{\lambda \in \mathbb{F}_2^n} \mathcal{F}^2(\Delta_a F_\lambda) \geq 2^{2n+1}.$$

Also, F is APN if and only if this equality holds.

Proof:

Let us consider the following value for any nonzero $a \in \mathbb{F}_2^n$.

$$\begin{aligned} A &= \sum_{\lambda \in \mathbb{F}_2^n} \mathcal{F}^2(\Delta_a F_\lambda) \\ &= \sum_{\lambda \in \mathbb{F}_2^n} \sum_{x \in \mathbb{F}_2^n} ((-1)^{(\lambda F(x+a) + \lambda F(x))})^2 \\ &= \sum_{\lambda \in \mathbb{F}_2^n} \sum_{x \in \mathbb{F}_2^n} \sum_{y \in \mathbb{F}_2^n} (-1)^{(\lambda(F(x+a) + F(x) + F(y+a) + F(y)))}. \end{aligned}$$

So we get

$$\begin{aligned} A &= 2^n \times \#\{(x, y) \in (\mathbb{F}_2^n)^2 : \Delta_a F(x) = \Delta_a F(y)\} \\ &= 2^{2n+1} + 2^n \times \#\{(x, y) \in (\mathbb{F}_2^n)^2 : \Delta_a F(x) = \Delta_a F(y), x \neq y \neq x+a\} \\ &= 2^{2n+1} + 2^n \times B, \text{ where } B = \#\{(x, y) \in (\mathbb{F}_2^n)^2 : \Delta_a F(x) = \Delta_a F(y), x \neq y \neq x+a\}. \end{aligned}$$

Thus obviously $A \geq 2^{2n+1}$ and from the definition of APN functions we can say F is an APN function if and only if $B = 0$ and so the equality holds. \square

This theorem is important for proving that F is not an APN function. Here we present one corollary of this theorem.

Definition 14 *The sum-of-square indicator of a Boolean function f is defined as*

$$\nu(f) = \sum_{a \in \mathbb{F}_2^n} \mathcal{F}^2(\Delta_a f).$$

Corollary 1 *(Corollary 1 of [2]) Let F be a function from \mathbb{F}_2^n to itself with components $F_\lambda, \lambda \in \mathbb{F}_2^n$. Then*

$$\sum_{\lambda \in \mathbb{F}_2^n \setminus \{0\}} \nu(F_\lambda) \geq (2^n - 1)2^{2n+1}.$$

Moreover, F is APN if and only if this equality holds.

Proof:

Let $A = \sum_{a \in \mathbb{F}_2^n \setminus \{0\}} \sum_{\lambda \in \mathbb{F}_2^n \setminus \{0\}} \mathcal{F}^2(\Delta_a F_\lambda)$. So from the previous theorem $A \geq 2^{2n+1}(2^n - 1)$. Again $\mathcal{F}^2(\Delta_0 F_\lambda) = \mathcal{F}^2(\Delta_a F_0) = 2^{2n}$ for any a and λ , then

$$\begin{aligned}
A &= \sum_{a \in \mathbb{F}_2^n \setminus \{0\}} \sum_{\lambda \in \mathbb{F}_2^n} \mathcal{F}^2(\Delta_a F_\lambda) \\
&= \sum_{a \in \mathbb{F}_2^n \setminus \{0\}} \sum_{\lambda \in \mathbb{F}_2^n \setminus \{0\}} \mathcal{F}^2(\Delta_a F_\lambda) + \sum_{a \in \mathbb{F}_2^n \setminus \{0\}} \mathcal{F}^2(\Delta_a F_0) \\
&= \sum_{\lambda \in \mathbb{F}_2^n \setminus \{0\}} \sum_{a \in \mathbb{F}_2^n \setminus \{0\}} \mathcal{F}^2(\Delta_a F_\lambda) + \sum_{\lambda \in \mathbb{F}_2^n \setminus \{0\}} \mathcal{F}^2(\Delta_0 F_\lambda) \\
&= \sum_{\lambda \in \mathbb{F}_2^n \setminus \{0\}} \sum_{a \in \mathbb{F}_2^n} \mathcal{F}^2(\Delta_a F_\lambda) \\
&= \sum_{\lambda \in \mathbb{F}_2^n \setminus \{0\}} \nu(F_\lambda).
\end{aligned}$$

Thus, $\sum_{\lambda \in \mathbb{F}_2^n \setminus \{0\}} \nu(F_\lambda) = A \geq (2^n - 1)2^{2n+1}$ and from the previous theorem F is APN if and only if $A = (2^n - 1)2^{2n+1}$. This completes the proof. \square

Here let us discuss some known monomial families of APN functions. Monomial families are of the form $F(x) = x^d$ over \mathbb{F}_{2^n} . For example

- Gold [16]: $F(x) = x^{2^t+1}$ where $\gcd(t, n) = 1$,
- $F(x) = x^3$ for any n (special case of Gold family for $t = 3$),
- Kasami [18] : $F(x) = x^{2^{2t}-2^t+1}$ where $\gcd(t, n) = 1$,
- Niho [13]: For $n = 2t + 1$,

$$F(x) = \begin{cases} x^{2^t+2^{\frac{t}{2}}-1}, & \text{for } t \text{ is even} \\ x^{2^t+2^{\frac{3t+1}{2}}-1}, & \text{for } t \text{ is odd} \end{cases}$$

- Welch [12]: For $n = 2t + 1$, $F(x) = x^{2^t+3}$,
- Dobbertin [14]: $F(x) = x^{2^{4t}+2^{3t}+2^{2t}+2^t-1}$ where $n = 5t$.

Also, there are polynomial APN families and all the infinite polynomial families are quadratic. However, our main objective is to find APN permutations and no such permutation is known for n even except Dillon's permutation over \mathbb{F}_2^6 , which is CCZ-equivalent[23] to a quadratic APN function. But when n is odd the following is a well-known APN permutation over \mathbb{F}_{2^n} [22]:

$$F(x) = \begin{cases} x^{-1}, & \text{for } x \neq 0 \\ 0, & \text{for } x = 0. \end{cases}$$

Non-Linearity

Linear cryptanalysis is a powerful method of cryptanalysis of block ciphers introduced by Matsui in 1994 [20]. Linear cryptanalysis is a known plaintext attack in which the attacker studies probabilistic linear relations (called linear approximations) between bits of the plaintext, the ciphertext, and the secret key. Like the *DDT* we can define the *Linear Approximation Table (LAT)*.

Definition 15 *The linear approximation table (LAT) of a vectorial Boolean function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is a $2^n \times 2^n$ matrix with entry at (a, b) position,*

$$\lambda_F(a, b) = \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot F(x)} = W_F(a, b)$$

for any $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^n$.

The efficiency of a linear attack depends on the values in the *LAT*. Again here a higher value helps the attacker.

Definition 16 *The linearity of a vectorial Boolean function F is*

$$\Lambda_F = \max_{a, b \neq 0 \in \mathbb{F}_2^n} |\lambda_F(a, b)| = \max_{a, b \neq 0 \in \mathbb{F}_2^n} |W_F(a, b)|.$$

On the other hand the non-linearity is

$$\eta_F = 2^{n-1} - \frac{\Lambda_F}{2}.$$

Also, there is a relation between the *DDT* and the *LAT*, described in the following proposition. This is a well-know result, which we proved for sake of completeness.

Proposition 2 *For a function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$*

$$\lambda_F(a, b)^2 = \sum_{\alpha \in \mathbb{F}_2^n} \sum_{\beta \in \mathbb{F}_2^n} \delta_F(\alpha, \beta) (-1)^{a \cdot \alpha + b \cdot \beta}, \text{ with } a, b \in \mathbb{F}_2^n.$$

Proof:

For any $a, b \in \mathbb{F}_2^n$ we have

$$\begin{aligned}
\lambda_F(a, b)^2 &= \left(\sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot F(x)} \right)^2 \\
&= \sum_{x \in \mathbb{F}_2^n} \sum_{y \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot F(x)} (-1)^{a \cdot y + b \cdot F(y)} \\
&= \sum_{x \in \mathbb{F}_2^n} \sum_{\alpha \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot F(x)} (-1)^{a \cdot (x + \alpha) + b \cdot F(x + \alpha)} \\
&= \sum_{x \in \mathbb{F}_2^n} \sum_{\alpha \in \mathbb{F}_2^n} (-1)^{a \cdot \alpha} (-1)^{b \cdot (F(x) + F(x + \alpha))} \\
&= \sum_{\alpha \in \mathbb{F}_2^n} (-1)^{a \cdot \alpha} \sum_{x \in \mathbb{F}_2^n} (-1)^{b \cdot (F(x) + F(x + \alpha))} \\
&= \sum_{\alpha \in \mathbb{F}_2^n} (-1)^{a \cdot \alpha} \sum_{\beta \in \mathbb{F}_2^n} (-1)^{b \cdot \beta} \delta_F(\alpha, \beta).
\end{aligned}$$

□

Here we will define one more relation between the DDT and the LAT from [11]. We will use the following lemmas.

Lemma 1 (*Lemma 1 of [11]*) For all $(a, b) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, we have

$$\delta_F(a, b) = (\theta_F \otimes \theta_F)(a, b).$$

Proof:

From the definition of characteristic function we have:

$$\begin{aligned}
(\theta_F \otimes \theta_F)(a, b) &= \sum_{x, y \in \mathbb{F}_2^n} \theta_F(x, y) \theta_F(a + x, b + y) \\
&= \sum_{x \in \mathbb{F}_2^n} \theta_F(a + x, b + F(x)) \\
&= \#\{x \in \mathbb{F}_2^n : F(a + x) = b + F(x)\} \\
&= \delta_F(a, b).
\end{aligned}$$

□

Lemma 2 (*Lemma 2 of [11]*) For all $(a, b) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, we have

$$\lambda_F(a, b) = \widehat{\theta}_F(a, b).$$

Proof:

From the definition of discrete Fourier transform we have:

$$\begin{aligned}\hat{\theta}_F(a, b) &= \sum_{x, y \in \mathbb{F}_2^n} \theta_F(x, y) (-1)^{a \cdot x + b \cdot y} \\ &= \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot F(x)} \\ &= \lambda_F(a, b).\end{aligned}$$

□

Lemma 3 For any function $f : \mathbb{F}_2^n \rightarrow \mathbb{R}$, we get

$$\hat{\hat{f}}(x) = 2^n f(x).$$

Proof:

We have:

$$\begin{aligned}\hat{\hat{f}}(x) &= \sum_{a \in \mathbb{F}_2^n} \hat{f}(a) (-1)^{a \cdot x} \\ &= \sum_{a \in \mathbb{F}_2^n} \sum_{b \in \mathbb{F}_2^n} f(b) (-1)^{a \cdot x + b \cdot a} \\ &= \sum_{a \in \mathbb{F}_2^n} \sum_{b \in \mathbb{F}_2^n} f(b) (-1)^{a \cdot (x+b)} \\ &= \sum_{a \in \mathbb{F}_2^n} f(x) \\ &= 2^n f(x).\end{aligned}$$

□

Theorem 3 For a function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$

$$\delta_F(a, b) = \frac{1}{2^{2n}} \widehat{\lambda}_F^2(a, b), \text{ with } a, b \in \mathbb{F}_2^n.$$

Proof:

We have:

$$\begin{aligned}\delta_F(a, b) &= (\theta_F \otimes \theta_F)(a, b), \text{ from lemma 1} \\ &= \frac{1}{2^{2n}} \widehat{(\theta_F \otimes \theta_F)}(a, b), \text{ from lemma 3} \\ &= \frac{1}{2^{2n}} \widehat{(\hat{\theta}_F)^2}(a, b) \\ &= \frac{1}{2^{2n}} \widehat{\lambda}_F^2(a, b), \text{ from lemma 2.}\end{aligned}$$

This completes the proof.

□

Bound on Linearity

If linearity is higher then the corresponding linear attack will be more efficient. So it is important to find a lower bound on the linearity.

Proposition 3 *If $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ is a Boolean function then*

$$\max_{a \in \mathbb{F}_2^n} |W_f(a)| \geq 2^{n/2}.$$

Proof : Suppose $\max_{a \in \mathbb{F}_2^n} |W_f(a)| < 2^{n/2}$ then from Parseval equality we got $2^{2n} = \sum_{a \in \mathbb{F}_2^n} (W_f(a))^2 < 2^n \times 2^n$, which is a contradiction.

If the equality holds then the function is called a *bent function*. Obviously *bent functions* exist only if n is even.

Now for a vectorial Boolean function, from the definition of the linearity

$$\Lambda_F = \max_{a, b \neq 0 \in \mathbb{F}_2^n} |W_F(a, b)| = \max_{a, b \neq 0 \in \mathbb{F}_2^n} |W_{F_b}(a)| \geq 2^{n/2},$$

where $F_b(x) = b \cdot F(x)$. Thus the above definition of *bent function* can be extended for vectorial Boolean functions. This has been extended by Nyberg [21].

Definition 17 *A vectorial Boolean function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ is bent if and only if, for all $b \in \mathbb{F}_2^m$, the Boolean function $F_b(x) = b \cdot F(x)$ is bent.*

Also from the definition of non-linearity we can deduce

$$\eta_F = 2^{n-1} - \frac{\Lambda_F}{2} \leq 2^{n-1} - 2^{\frac{n}{2}-1}.$$

Let us state an important theorem on the existence of bent function from [11].

Theorem 4 *Let us consider a function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$. Then F is bent only if $n \geq 2m$ and n is even.*

From this theorem we can say that for vectorial Boolean functions from \mathbb{F}_2^n to itself, bent functions do not exist. The smallest linearity is $2^{\frac{n+1}{2}}$ and the functions F which satisfy $\Lambda(F) = 2^{\frac{n+1}{2}}$ are said to be *almost bent* (AB). They exist only when n is odd. Moreover, if F is AB then

$$\{W_F(a, b) : a \in \mathbb{F}_2^n, b \in \mathbb{F}_2^n \setminus \{0\}\} = \{0, \pm 2^{\frac{n+1}{2}}\}.$$

Now we describe an important result on the following theorem.

Theorem 5 ([11]) *If the function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is almost bent (AB) then F is almost perfect nonlinear (APN).*

But the converse of the above theorem does not hold in general. But for quadratic functions, APN implies AB [10].

Vector Spaces Extraction

In this section we will provide a mathematical tool to find all the vector spaces of dimension at least, say, d that are completely contained in some set $S \in \mathbb{F}_2^n$. Most of this part is from the paper [4]. Indeed in Chapter 3 we will provide a process to find a permutation from a function and in that construction we will require this tool for finding vector space contained in a special set. Besides we will provide an algorithm to find the rank of a Boolean matrix efficiently because in our main algorithms in Chapter 4, we need to determine the rank of a Boolean matrix a huge number of times. We managed both efficiently using the same framework.

Definition 18 *Let $x = (x[0], x[1], \dots, x[n-1]) \in \mathbb{F}_2^n$. Then $x[0]$ is the least significant bit and $MSB(x)$ is the greatest index i such that $x[i] = 1$.*

Lemma 4 *For any $x, y \in \mathbb{F}_2^n$, it holds that*

$$x < x \oplus y \Leftrightarrow x[MSB(y)] = 0,$$

where \oplus is bitwise xor and the order is the ordering of integers obtained by interpreting $(x[0], x[1], \dots, x[n-1])$ as the binary representation of an integer.

Let us describe an efficient rank finding algorithm using this lemma. This algorithm takes as input a Boolean matrix and the size of that matrix and outputs the rank. The idea of this algorithm is the same as of Gaussian elimination. If M_i is the integer representation of the i th row of the matrix M , then by the above lemma $M_i \oplus M_j < M_j$ implies that $M_j[MSB(M_i)] = 1$. So we will replace M_j by $M_i \oplus M_j$ and once it completes iterations through all j s, we will check whether M_i is non-zero or not. The subroutine Rank is specified in Algorithm 2.4.1.

Algorithm 2.4.1: RANK($M, SIZE$)

```

rank ← 0
for i ← 0 to SIZE - 1
  do {
    for j ← i + 1 to SIZE - 1
      do {
        Z ← Mi ⊕ Mj
        if (Z < Mj)
          then Mj ← Z
      }
    if (Mi > 0)
      then rank ← rank + 1
  }
return (rank)

```

Now let us come back to our main discussion on vector space extraction. A naive approach to construct such a vector space is as follows. We will first choose

an element $x \in S$ and construct the vector space $V_x = \{0, x\}$. Then we will choose all the elements y one by one such that $y > x$ and $x \oplus y \in S$ and will construct $V_{x,y} = V_x \cup \{y \oplus v, v \in V_x\}$. We repeat this process until we get a basis $\{x, y, z, \dots\}$ of the required size. But the main problem with this approach is that the basis of a vector space is not unique. So if we approach with the above process the algorithm will be slowed down to explore all of the branches that lead to the basis of the same vector space. The process in [4] solves this problem considering a unique basis, called *Gauss-Jordan Basis (GJB)*.

Definition 19 (*Gauss-Jordan Basis (GJB)*) For any vector space V of dimension d , the *Gauss-Jordan Basis (GJB)* of V is the set $\{v_0, v_1, \dots, v_{d-1}\}$ such that

$$\text{span}\{v_0, v_1, \dots, v_{d-1}\} = V$$

and which is the smallest set when stored in lexicographic order.

Let us discuss some properties of GJBs form [4].

Lemma 5 If $\{v_0, v_1, \dots, v_{d-1}\}$ is the GJB of $\text{span}\{v_0, v_1, \dots, v_{d-1}\}$ then $\{v_0, v_1, \dots, v_{d-2}\}$ is also the GJB of its span.

Proof:

If $\{v_0, v_1, \dots, v_{d-2}\}$ is not a GJB then we can find a basis of $\text{span}\{v_0, v_1, \dots, v_{d-2}\}$ which can be used to find a basis of $\text{span}\{v_0, v_1, \dots, v_{d-1}\}$ lexicographically smaller than $\{v_0, v_1, \dots, v_{d-1}\}$. Which is a contradiction as $\{v_0, v_1, \dots, v_{d-1}\}$ is a GJB. \square

Lemma 6 The basis $\{v_0, v_1, \dots, v_{d-1}\}$ is a GJB if and only if the following two conditions hold.

$$\begin{cases} \forall j \in \{0, 1, \dots, d-2\}, \text{MSB}(v_j) < \text{MSB}(v_{j+1}) \\ \forall i \in \{1, 2, \dots, d-1\}, \forall j \in \{0, 1, \dots, i-1\}, v_i[\text{MSB}(v_j)] = 0. \end{cases}$$

Proof:

\Rightarrow Suppose $\{v_0, v_1, \dots, v_{d-1}\}$ is a GJB. If $\text{MSB}(v_j) = \text{MSB}(v_{j+1})$ then $v_j \oplus v_{j+1} < v_j$, which contradicts the fact that $\{v_0, v_1, \dots, v_{d-1}\}$ is a GJB. We get a similar contradiction if $\text{MSB}(v_j) < \text{MSB}(v_{j+1})$ as this imply $v_{j+1} < v_j$. So we deduced that $\text{MSB}(v_j) < \text{MSB}(v_{j+1})$ for any $j \in \{0, 1, \dots, d-2\}$.

Suppose $v_i[\text{MSB}(v_j)] = 1$ for some $j < i$. Then from Lemma 4 we get $v_i \geq v_i \oplus v_j$. This is again a contradiction as $\{v_0, v_1, \dots, v_{d-1}\}$ is a GJB.

\Leftarrow Conversely let us assume that the conditions hold. Then for any subset I of $\{0, 1, \dots, i-1\}$, the *MSB* of $\bigoplus_{j \in I} v_j$ is always strictly smaller than $\text{MSB}(v_i)$. Then again from Lemma 4 and the second condition $v_i < (\bigoplus_{j \in I} v_j) \oplus v_i$. Thus adding v_i to the end of a GJB $\{v_0, v_1, \dots, v_{i-1}\}$ yields a GJB. So using an induction we can prove the result. \square

Lemma 7 *If $\{v_0, v_1, \dots, v_{d-1}\}$ is a GJB then, $\forall j \in \{0, 1, \dots, d-1\}$, $\text{span}\{v_0, v_1, \dots, v_{d-1}\}$ contains exactly 2^j elements x such that $\text{MSB}(x) = \text{MSB}(v_j)$.*

Proof:

We can prove this using induction over the size of the basis. If the basis is simply $\{v_0\}$ then this lemma obviously holds. Now adding an element v_d to the end of a GJB of size d will add 2^d new elements x such that $\text{MSB}(x) = \text{MSB}(v_d)$. \square

Definition 20 (*MSB spectrum*) *The MSB spectrum of a set $S \subseteq \mathbb{F}_2^n$ is the sequence $\{N_i(S)\}_{0 \leq i < n}$ such that*

$$N_i(S) = \#\{x \in S : \text{MSB}(x) = i\}.$$

From this concept of MSB spectrum we can get the following corollary of Lemma 7.

Corollary 2 (*MSB condition*) *If a set $S \subseteq \mathbb{F}_2^n$ contains a vector space of dimension d , then there exists a strictly increasing sequence $\{m_j\}_{0 \leq j < d}$ of length d such that $N_{m_j}(S) \geq 2^j$.*

Definition 21 (*Extraction [4]*) *For any nonzero element a of \mathbb{F}_2^n , the extraction of a , denoted by χ_a is a map mapping a subset S of \mathbb{F}_2^n to $\chi_a(S)$, where $x \in \chi_a(S)$ if and only if all of the following conditions are satisfied:*

$$x \in S, \quad x \oplus a \in S, \quad a < x < x \oplus a.$$

Vector space extraction will iterate such an extraction to construct vector spaces. The following theorem from [4] provides the main idea.

Theorem 6 *Let $\{v_0, v_1, \dots, v_{i-1}\}$ be elements of some subset S of \mathbb{F}_2^n such that $0 \in S$ and such that $v_{j+1} \in (\chi_{v_j} \circ \dots \circ \chi_{v_0})(S)$ for all $j < i$. Then it holds that $v_i \in (\chi_{v_{i-1}} \circ \dots \circ \chi_{v_0})(S)$ if and only if $\mathcal{L}(v_0, \dots, v_i) \subseteq S$ and $\{v_0, \dots, v_i\}$ is the GJB of this vector space.*

Corollary 3 *If $\{e_0, \dots, e_{d-1}\}$ is the GJB of a vector space $V \subseteq S \subseteq \mathbb{F}_2^n$ then, for all $0 < j < d$, we have*

$$\text{span}\{e_j, e_{j+1}, \dots, e_{d-1}\} \subseteq (\chi_{e_{j-1}} \circ \dots \circ \chi_{e_0})(S).$$

Thus we can check whether $x \oplus a \in S$ for all $x \in S$ such $x < x \oplus a$.

Now the following lemma provides a stronger filter. With this we can check whether an element can be the first element of a GJB.

Lemma 8 (*Bigger MSB condition [4]*) *If e_0 is the first element in a GJB of size d of a vector space $V \subseteq S \subseteq \mathbb{F}_2^n$, then let us define*

$$S' = \{x \in S : \text{MSB}(x) > \text{MSB}(e_0)\}.$$

Then S' must satisfy the MSB condition from Corollary 2 for dimension $d - 1$, i.e. there is a strictly increasing sequence m_j of length $d - 1$ such that

$$|\{x \in S : \text{MSB}(x) = m_j\}| > 2^j.$$

Thus using this lemma we can check if e_0 can be the first element of a GJB or not. Let ϕ_d be a mapping that maps a set S to $\phi_d(S)$ where

$$a \in \phi_d(S) \iff \exists \{m_j\}_{0 \leq j < d}, \begin{cases} m_{j+1} > m_j > MSB(a), \\ \#\{x \in S : MSB(x) = m_j\} > 2^j. \end{cases}$$

So $a \in \phi_d(S)$ implies a can be the first element of a GJB of size d . So the main algorithm is as follows.

Algorithm 2.4.2: GJBEXTRACTION(S, d)

```

List ← {}
for each  $a \in \phi_d(S)$ 
  do  $\begin{cases} s_a \leftarrow \chi_a(S) \\ \text{if } (|s_a| \geq 2^{d-1} - 1) \\ \text{then } \begin{cases} List_1 \leftarrow \text{GJBEXTRACTION}(s_a, \max(d-1, 0)) \\ \text{for each } B \in List_1 \\ \text{do Add the GJB}(\{a\} \cup B) \text{ to List} \end{cases} \end{cases}$ 
return (List)

```

Chapter 3

CCZ-Equivalence

In this chapter we will discuss CCZ-equivalence. Two vectorial Boolean functions are CCZ-equivalent if there is an affine permutation that maps the graph of one function to another. This class is important because many cryptographic properties are the same for every member inside a class. So analysing one member of the class is enough to know about its other members. We also discuss EA-equivalence which is a particular case of CCZ-equivalence. Most of the definitions and results of this chapter are taken from [9].

This chapter provides the main mathematical motivation of our work as our approach will be try to find an APN function which is “CCZ-equivalent” to a permutation. We will start with the definition and properties of CCZ-equivalence in Section 3.1. In Section 3.2 we will describe how the DDT and LAT of two functions in the same CCZ-equivalence class are related. In Section 3.3 we will describe two special kinds of sets called vector space of zeros. Finally we finish this chapter with our main discussion on constructing a CCZ-equivalent permutation from a function in Section 3.4.

CCZ-Equivalence and EA-Equivalence

Definition 22 Consider two functions $F, G : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. F and G are Extended-affine equivalent (EA-equivalent) if there exist two affine permutations $A, B : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and an affine function $C : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ such that

$$F(x) = (B \circ G \circ A)(x) + C(x).$$

If C is a zero map then we call F and G affine equivalent.

Definition 23 Two functions $F, G : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ are CCZ-equivalent if there exists an affine permutation $\mathcal{A} : \mathbb{F}_2^{2n} \rightarrow \mathbb{F}_2^{2n}$ such that

$$\{(x, F(x)) : x \in \mathbb{F}_2^n\} = \mathcal{A}(\{(x, G(x)) : x \in \mathbb{F}_2^n\}).$$

CCZ-equivalence was named after Carlet, Charpin and Zinoviev who first introduced this in [10].

Definition 24 *The graph of a function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, denoted by Γ_F , is the set*

$$\Gamma_F = \{(x, F(x)) : x \in \mathbb{F}_2^n\}.$$

So F and G are CCZ-equivalent if there exists an affine permutation \mathcal{A} such that $\Gamma_F = \mathcal{A}(\Gamma_G)$.

Lemma 9 *EA-equivalence is a particular case of CCZ-equivalence.*

Proof:

Consider two functions $F, G : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Suppose F and G are EA-equivalent. Let $F(x) = (B \circ G \circ A)(x) + C(x)$. Now from this we write

$$\begin{aligned} \left\{ \begin{bmatrix} x \\ F(x) \end{bmatrix} : x \in \mathbb{F}_2^n \right\} &= \left\{ \begin{bmatrix} A^{-1}A(x) \\ BGA(x) + C(x) \end{bmatrix} : x \in \mathbb{F}_2^n \right\} \\ &= \left\{ \begin{bmatrix} A^{-1} & 0 \\ CA^{-1} & B \end{bmatrix} \times \begin{bmatrix} A(x) \\ G(A(x)) \end{bmatrix} : x \in \mathbb{F}_2^n \right\} \\ &= \left\{ \begin{bmatrix} A^{-1} & 0 \\ CA^{-1} & B \end{bmatrix} \times \begin{bmatrix} y \\ G(y) \end{bmatrix} : y \in \mathbb{F}_2^n \right\}. \end{aligned}$$

So we get $\Gamma_F = \mathcal{M}(\Gamma_G)$, where

$$\mathcal{M} = \begin{bmatrix} A^{-1} & 0 \\ CA^{-1} & B \end{bmatrix}.$$

Thus we can say F and G are CCZ-equivalent. \square

EL-mapping and EA-mapping

Definition 25 *A linear permutation from $\mathbb{F}_2^n \times \mathbb{F}_2^n$ to itself is called an EL-mapping if its matrix representation is of the form*

$$\begin{bmatrix} A & 0 \\ B & C \end{bmatrix}.$$

Where $A, C : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ are linear permutation and $B : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is a linear function.

An affine permutation from $\mathbb{F}_2^n \times \mathbb{F}_2^n$ to itself is called an EA-mapping if its linear part is an EL-mapping.

With this definition we can get a definition of EA-equivalence similar to CCZ-equivalence. Two functions F and G are EA-equivalent if and only if there is an EA-mapping such that $\Gamma_F = \mathcal{A}(\Gamma_G)$. The ‘only if’ part is already proved in Lemma 9.

We will consider two orthogonal spaces $\nu = \{(x, 0) : x \in \mathbb{F}_2^n\}$ and $\nu^\perp = \{(0, x) : x \in \mathbb{F}_2^n\}$.

Lemma 10 *A linear map $\mathcal{A} : \mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n \times \mathbb{F}_2^n$ is an EL-mapping if and only if $\mathcal{A}^T(\nu) = \nu$.*

Also an affine map $\mathcal{A} : \mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n \times \mathbb{F}_2^n$ with linear part \mathcal{L} is an EA-mapping if and only if $\mathcal{L}^T(\nu) = \nu$.

Admissible Affine Permutation

If a function F and an affine permutation \mathcal{A} are given then the existence of G such that $\mathcal{A}(\Gamma_F) = \Gamma_G$ is not guaranteed. Those permutations corresponding to a function F , for which such a G exists, are important. So we can get the following definition.

Definition 26 *An affine permutation $\mathcal{A} : \mathbb{F}_2^{2n} \rightarrow \mathbb{F}_2^{2n}$ is admissible if we can define a function G such that $\mathcal{A}(\Gamma_F) = \Gamma_G$.*

One important admissible mapping corresponding to any permutation P is

$$\mathcal{L} = \begin{bmatrix} 0 & I_n \\ I_n & 0 \end{bmatrix}$$

which maps the graph of P to the graph of P^{-1} .

Proposition 4 *If $\mathcal{A}(x, y) = (\mathcal{A}_1(x, y), \mathcal{A}_2(x, y))$ with $\mathcal{A}_1 : \mathbb{F}_2^{2n} \rightarrow \mathbb{F}_2^n$ and $\mathcal{A}_2 : \mathbb{F}_2^{2n} \rightarrow \mathbb{F}_2^n$ then \mathcal{A} is admissible if and only if $x \mapsto \mathcal{A}_1(x, y)$ is a permutation of \mathbb{F}_2^n for all $y \in \mathbb{F}_2^n$.*

Proof:

If \mathcal{A} is admissible for F then there is some G such that $\mathcal{A}(\Gamma_F) = \Gamma_G$. Thus we write

$$\left\{ \left[\begin{array}{c} \mathcal{A}_1(x, F(x)) \\ \mathcal{A}_2(x, F(x)) \end{array} \right] : x \in \mathbb{F}_2^n \right\} = \left\{ \left[\begin{array}{c} x \\ G(x) \end{array} \right] : x \in \mathbb{F}_2^n \right\}.$$

Hence \mathcal{A}_1 is a permutation of \mathbb{F}_2^n .

Conversely, let \mathcal{A}_1 be a permutation of \mathbb{F}_2^n . Let us define two functions $F_1 : x \mapsto \mathcal{A}_1(x, F(x))$ and $F_2 : x \mapsto \mathcal{A}_2(x, F(x))$. Then F_1 is a permutation. So we define a

function $G(x) = F_2 \circ F_1^{-1}(x) = \mathcal{A}_2(F_1^{-1}(x), F \circ F_1^{-1}(x))$. Thus we get the following:

$$\begin{aligned}
 \mathcal{A}(\Gamma_F) &= \left\{ \left[\begin{array}{c} \mathcal{A}_1(x, F(x)) \\ \mathcal{A}_2(x, F(x)) \end{array} \right] : x \in \mathbb{F}_2^n \right\} \\
 &= \left\{ \left[\begin{array}{c} F_1(x) \\ \mathcal{A}_2(x, F(x)) \end{array} \right] : x \in \mathbb{F}_2^n \right\} \\
 &= \left\{ \left[\begin{array}{c} y \\ \mathcal{A}_2(F_1^{-1}(y), F \circ F_1^{-1}(y)) \end{array} \right] : y \in \mathbb{F}_2^n \right\} \\
 &= \left\{ \left[\begin{array}{c} y \\ G(y) \end{array} \right] : y \in \mathbb{F}_2^n \right\} \\
 &= \Gamma_G.
 \end{aligned}$$

This proves that \mathcal{A} is admissible. \square

So an EA-mapping is always admissible. Thus for any function F and EA-mapping \mathcal{A} , we can get a function G which is EA-equivalent to F . But there are functions for which we can get admissible permutation which are not EA-mappings (as in the above example of admissible mapping which maps the graph of P to the graph of P^{-1}). So CCZ-equivalence is more general than EA-equivalence.

DDT and LAT of CCZ Equivalent Functions

Now we will discuss about the DDT and LAT of functions in the same CCZ-equivalence class.

Proposition 5 (Proposition 2 of [9]) *If $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and $G : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ are CCZ-equivalent due to the affine permutation \mathcal{A} where $\mathcal{A}(x) = \mathcal{L}(x) + c$, then the relation between the DDT of the two functions is*

$$\delta_G(a, b) = \delta_F(\mathcal{L}^{-1}(a, b))$$

and the relation between their LAT is

$$\lambda_G(a, b) = (-1)^{c \cdot (a, b)} \lambda_F(\mathcal{L}^T(a, b)).$$

Proof:

From the definition of DDT we have

$$\begin{aligned}
 \delta_G(a, b) &= \#\{x \in \mathbb{F}_2^n : G(x+a) + G(x) = b\} \\
 &= \#\{(x, x') \in \mathbb{F}_2^n \times \mathbb{F}_2^n : G(x) + G(x') = b \text{ and } x + x' = a\}.
 \end{aligned}$$

Now let $\mathcal{A}(x, y) = (\mathcal{A}_1(x, y), \mathcal{A}_2(x, y))$ where $\mathcal{A}_1(x) = \mathcal{L}_1(x) + c_1$ and $\mathcal{A}_2(x) = \mathcal{L}_2(x) + c_2$ i.e the linear part of \mathcal{A}_1 is \mathcal{L}_1 and linear part of \mathcal{A}_2 is \mathcal{L}_2 and $c = (c_1, c_2)$. Also let us take $F_1 : x \mapsto \mathcal{A}_1(x, F(x))$ and $F_2 : x \mapsto \mathcal{A}_2(x, F(x))$ as in Proposition 4. So F_1 is a permutation and $G = F_2 \circ F_1^{-1}$.

Now if we take $y = F_1^{-1}(x)$ and $y' = F_1^{-1}(x')$, we get

$$\begin{aligned}
\delta_G(a, b) &= \#\{(y, y') \in \mathbb{F}_2^n \times \mathbb{F}_2^n : F_2(y) + F_2(y') = b \text{ and } F_1(y) + F_1(y') = a\} \\
&= \#\{(y, y') \in \mathbb{F}_2^n \times \mathbb{F}_2^n : \mathcal{A}_2(y, F(y)) + \mathcal{A}_2(y', F(y')) = b \text{ and} \\
&\quad \mathcal{A}_1(y, F(y)) + \mathcal{A}_1(y', F(y')) = a\} \\
&= \#\{(y, y') \in \mathbb{F}_2^n \times \mathbb{F}_2^n : \mathcal{L}_2(y, F(y)) + \mathcal{L}_2(y', F(y')) = b \text{ and} \\
&\quad \mathcal{L}_1(y, F(y)) + \mathcal{L}_1(y', F(y')) = a\} \\
&= \#\{(y, y') \in \mathbb{F}_2^n \times \mathbb{F}_2^n : \mathcal{L}_2(y + y', F(y) + F(y')) = b \text{ and} \\
&\quad \mathcal{L}_1(y + y', F(y) + F(y')) = a\} \\
&= \#\{(y, y') \in \mathbb{F}_2^n \times \mathbb{F}_2^n : \mathcal{L}(y + y', F(y) + F(y')) = (a, b)\} \\
&= \#\{(y, y') \in \mathbb{F}_2^n \times \mathbb{F}_2^n : (y + y', F(y) + F(y')) = \mathcal{L}^{-1}(a, b)\} \\
&= \delta_F(\mathcal{L}^{-1}(a, b))
\end{aligned}$$

This completes the proof of the first part.

For the second part let us recall that

$$\lambda_G(a, b) = \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot G(x)},$$

so $\lambda_G(a, b)$ can be determined by

$$\#\{x \in \mathbb{F}_2^n : b \cdot G(x) + a \cdot x = 0\}.$$

Thus from the left hand side we get

$$\begin{aligned}
b \cdot G(x) + a \cdot x &= b \cdot F_2(y) + a \cdot F_1(y) \text{ where } y = F_1^{-1}(x) \\
&= b \cdot \mathcal{A}_2(y, F(y)) + a \cdot \mathcal{A}_1(y, F(y)) \\
&= b \cdot \mathcal{L}_2(y, F(y)) + b \cdot c_2 + a \cdot \mathcal{L}_1(y, F(y)) + a \cdot c_1 \\
&= b \cdot \mathcal{L}_2(y, F(y)) + a \cdot \mathcal{L}_1(y, F(y)) + c \cdot (a, b) \\
&= (a, b) \cdot \mathcal{L}(y, F(y)) + c \cdot (a, b) \\
&= \mathcal{L}^T(a, b) \cdot (y, F(y)) + c \cdot (a, b).
\end{aligned}$$

So finally,

$$\begin{aligned}
\lambda_G(a, b) &= \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot G(x)} \\
&= \sum_{y \in \mathbb{F}_2^n} (-1)^{\mathcal{L}^T(a, b) \cdot (y, F(y)) + c \cdot (a, b)} \\
&= (-1)^{c \cdot (a, b)} \sum_{y \in \mathbb{F}_2^n} (-1)^{\mathcal{L}^T(a, b) \cdot (y, F(y))} \\
&= (-1)^{c \cdot (a, b)} \lambda_F(\mathcal{L}^T(a, b)).
\end{aligned}$$

□

Vector Space of Zeros

Here we will find the condition for a map to be admissible using two special kinds of sets.

Definition 27 (*Walsh zeroes*) Let us consider $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Then the Walsh zeroes of F is the set

$$\mathcal{Z}_F = \{(a, b) \in \mathbb{F}_2^{2n} : \lambda_F(a, b) = 0\} \cup (0, 0).$$

Definition 28 (*Impossible differential set*) Let us consider $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Then the impossible differential set of F is the set

$$\mathcal{Z}_F^D = \{(a, b) \in \mathbb{F}_2^{2n} : \delta_F(a, b) = 0\} \cup (0, 0).$$

The relation between these two sets of zeroes is in the following proposition.

Proposition 6 (*Proposition 3 of [9]*) Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. For any subspace $V \subset \mathbb{F}_2^{2n}$ of dimension n , $V \subseteq \mathcal{Z}_F \Leftrightarrow V^\perp \subseteq \mathcal{Z}_F^D$.

Proof:

Recall the relation between DDT and LAT from Proposition 2.

$$(\lambda_F(a, b))^2 = \sum_{\alpha \in \mathbb{F}_2^n} \sum_{\beta \in \mathbb{F}_2^n} \delta_F(\alpha, \beta) (-1)^{a \cdot \alpha + b \cdot \beta}, \text{ with } a, b \in \mathbb{F}_2^n.$$

Now if we sum over the set V , we get,

$$\begin{aligned}
\sum_{(a, b) \in V} (\lambda_F(a, b))^2 &= \sum_{(a, b) \in V} \sum_{\alpha \in \mathbb{F}_2^n} \sum_{\beta \in \mathbb{F}_2^n} \delta_F(\alpha, \beta) (-1)^{a \cdot \alpha + b \cdot \beta} \\
&= \sum_{\alpha \in \mathbb{F}_2^n} \sum_{\beta \in \mathbb{F}_2^n} \delta_F(\alpha, \beta) \left(\sum_{(a, b) \in V} (-1)^{a \cdot \alpha + b \cdot \beta} \right).
\end{aligned}$$

Now

$$\sum_{(a,b) \in V} (-1)^{a \cdot \alpha + b \cdot \beta} = \begin{cases} 2^{\dim(V)}, & \text{if } (\alpha, \beta) \in V^\perp \\ 0, & \text{otherwise.} \end{cases}$$

So from the above equation we get,

$$\sum_{(a,b) \in V} (\lambda_F(a, b))^2 = 2^{\dim(V)} \sum_{(\alpha, \beta) \in V^\perp} \delta_F(\alpha, \beta).$$

Obviously, $V \subseteq \mathcal{Z}_F$ if and only if

$$\sum_{(a,b) \in V} (\lambda_F(a, b))^2 = (\lambda_F(0, 0))^2 = (2^n)^2 = 2^{2n}.$$

Here it is given that $\dim(V) = n$. So, $V \subseteq \mathcal{Z}_F$ if and only if $\sum_{(\alpha, \beta) \in V^\perp} \delta_F(\alpha, \beta) = 2^n = \delta_F(0, 0)$ which is equivalent to $V^\perp \subseteq \mathcal{Z}_F^D$. This completes the proof. \square

The following lemma describes the relation between the Walsh zeroes of two CCZ-equivalent functions.

Lemma 11 *If F and G are CCZ-equivalent due to the affine permutation \mathcal{A} i.e $\Gamma_G = \mathcal{A}(\Gamma_F)$, where $\mathcal{A}(x) = \mathcal{L}(x) + c$ then $\mathcal{Z}_G = (\mathcal{L}^T)^{-1}(\mathcal{Z}_F)$ and $\mathcal{Z}_G^D = \mathcal{L}(\mathcal{Z}_F^D)$.*

Proof:

From Proposition 5 we have the following relations:

$$\delta_G(a, b) = \delta_F(\mathcal{L}^{-1}(a, b)), \quad (3.1)$$

$$\lambda_G(a, b) = (-1)^{c \cdot (a, b)} \lambda_F(\mathcal{L}^T(a, b)). \quad (3.2)$$

If we take $(a, b) \in \mathcal{Z}_G$ then $\lambda_G(a, b) = 0$. So, from Equation (3.2), $\lambda_F(\mathcal{L}^T(a, b)) = 0$. Therefore from the definition of \mathcal{Z}_F we can write $\mathcal{L}^T(a, b) \in \mathcal{Z}_F$ and then $(a, b) \in (\mathcal{L}^T)^{-1}(\mathcal{Z}_F)$. So we get $\mathcal{Z}_G \subseteq (\mathcal{L}^T)^{-1}(\mathcal{Z}_F)$. Similarly we can show that $(\mathcal{L}^T)^{-1}(\mathcal{Z}_F) \subseteq \mathcal{Z}_G$ using the same Equation (3.2).

Again, for the second part let $(a, b) \in \mathcal{Z}_G^D$. Then we have $\delta_G(a, b) = 0$. Therefore from Equation (3.1) we get $\delta_F(\mathcal{L}^{-1}(a, b)) = 0$ and so $(a, b) \in \mathcal{L}(\mathcal{Z}_F^D)$. So finally we get $\mathcal{Z}_G^D \subseteq \mathcal{L}(\mathcal{Z}_F^D)$. Similarly we can prove for the other direction. \square

Let us recall that $\nu = \{(x, 0) : x \in \mathbb{F}_2^n\}$ and $\nu^\perp = \{(0, x) : x \in \mathbb{F}_2^n\}$.

Proposition 7 (Proposition 4 of [9]) *For any function F it always holds that*

$$\nu \subseteq \mathcal{Z}_F,$$

$$\nu^\perp \subseteq \mathcal{Z}_F^D.$$

Proof:

If $(a, b) \in \nu$ then $b = 0$. Now if $a = 0$ then by definition $(a, b) \in \mathcal{Z}_F$. Also, for all (a, b) with $a \neq 0$,

$$\lambda_F(a, b) = \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x + 0 \cdot F(x)} = \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x} = 0.$$

The last equality follows from the fact that if $a = (a_0, a_1, \dots, a_{n-1}) \neq 0$ and $x = (x_0, x_1, \dots, x_{n-1})$, then we get the following equation

$$a \cdot x = 0,$$

which is equivalent to

$$a_0 x_0 + \dots + a_{n-1} x_{n-1} = 0.$$

The above equation has exactly 2^{n-1} solutions. So the set $\{a \cdot x : x \in \mathbb{F}_2^n\}$ takes the value 0 and 1 the same number of times. \square

Now we present the main theorem of this section.

Theorem 7 (Theorem 1 of [9]) *Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and let \mathcal{A} be an affine permutation on $\mathbb{F}_2^n \times \mathbb{F}_2^n$ with linear part \mathcal{L} . Then the following statements are equivalent:*

1. \mathcal{A} is admissible for F ,
2. $\mathcal{L}^T(\nu) \subseteq \mathcal{Z}_F$,
3. $\mathcal{L}^{-1}(\nu^\perp) \subseteq \mathcal{Z}_F^D$.

Proof:

$1 \Leftrightarrow 2$. Let $\mathcal{A}(x, y) = \mathcal{L}(x, y) + c$ where $\mathcal{L}(x, y) = (\mathcal{L}_1(x, y), \mathcal{L}_2(x, y))$. Then, by Proposition 4, \mathcal{A} is admissible for F if and only if $x \mapsto \mathcal{L}_1(x, F(x))$ is a permutation. Again, the function $x \mapsto \mathcal{L}_1(x, F(x))$ is a permutation if and only if each of its components are balanced i.e

$$\sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot \mathcal{L}_1(x, F(x))} = 0, \forall a \in \mathbb{F}_2^n \setminus \{0\}.$$

So we get

$$\begin{aligned} & \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot \mathcal{L}_1(x, F(x))} = 0, \forall a \in \mathbb{F}_2^n \setminus \{0\} \\ \Leftrightarrow & \sum_{x \in \mathbb{F}_2^n} (-1)^{(a,0) \cdot (\mathcal{L}_1(x, F(x)), \mathcal{L}_2(x, F(x)))} = 0, \forall a \in \mathbb{F}_2^n \setminus \{0\} \\ \Leftrightarrow & \sum_{x \in \mathbb{F}_2^n} (-1)^{(a,0) \cdot \mathcal{L}(x, F(x))} = 0, \forall a \in \mathbb{F}_2^n \setminus \{0\} \\ \Leftrightarrow & \sum_{x \in \mathbb{F}_2^n} (-1)^{\mathcal{L}^T(a,0) \cdot ((x, F(x)))} = 0, \forall a \in \mathbb{F}_2^n \setminus \{0\}. \end{aligned}$$

Which is equivalent to $\mathcal{L}^T(\nu) \subseteq \mathcal{Z}_F$.

2 \Leftrightarrow 3. To prove this we need the following result.

Claim: For any vector space V , $\mathcal{L}^T(V)^\perp = \mathcal{L}^{-1}(V^\perp)$.

Proof of the claim:

$$\begin{aligned} x \in (\mathcal{L}^T(V))^\perp &\Leftrightarrow x \cdot \mathcal{L}^T(v) = 0, \forall v \in V \\ &\Leftrightarrow \mathcal{L}(x) \cdot v = 0, \forall v \in V \\ &\Leftrightarrow \mathcal{L}(x) \in V^\perp \\ &\Leftrightarrow x \in \mathcal{L}^{-1}(V^\perp). \end{aligned}$$

Now from Proposition 6, $\mathcal{L}^T(\nu) \subseteq \mathcal{Z}_F$ if and only if $(\mathcal{L}^T(\nu))^\perp \subseteq \mathcal{Z}_F^D$ and so from the above claim, $\mathcal{L}^{-1}(\nu^\perp) \subseteq \mathcal{Z}_F^D$. \square

TU Projection of a Function

Here we will discuss a mathematical concept to decompose a function in two parts. With this decomposition we will be able to find a condition for a function to be CCZ-equivalent to a permutation, which is the main object of this chapter.

Definition 29 (*Swap Matrix*) The t -Swap Matrix (for $t \leq n$) is a $2n$ -bit permutation, defined as

$$M_t = \begin{bmatrix} 0 & 0 & I_t & 0 \\ 0 & I_{n-t} & 0 & 0 \\ I_t & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{n-t} \end{bmatrix}$$

So $M_t \times (a, b, c, d)^T = (c, b, a, d)^T$ i.e it swaps a and c . Also, we can write $M_t = M_t^T = M_t^{-1}$. A special swap matrix for $t = n$ is

$$M_n = \begin{bmatrix} 0 & I_n \\ I_n & 0 \end{bmatrix}$$

So in this example $M_n(\nu) = \nu^\perp$. Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and t be an integer such that $0 \leq t \leq n$ then F can be written as $F(x, y) = (T_y(x), U_x(y))$, where

$$\begin{cases} \forall y \in \mathbb{F}_2^{n-t}, T_y : \mathbb{F}_2^t \rightarrow \mathbb{F}_2^t \\ \forall x \in \mathbb{F}_2^t, U_x : \mathbb{F}_2^{n-t} \rightarrow \mathbb{F}_2^{n-t}. \end{cases}$$

We call this TU-projection of F for t and this can be denoted as (T, U) . An important property of TU-projection is in the following proposition.

Proposition 8 (Proposition 6 of [9]) Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and $F(x, y) = (T_y(x), U_x(y))$ for some integer $t \leq n$.

Then T_y is permutation $\forall y$ if and only if $M_t(\nu^\perp) \subseteq \mathcal{Z}_F^D$, where M_t is the t -swap matrix.

Proof:

T_y is permutation $\forall y \in \mathbb{F}_2^{n-t}$ if and only if

$$T_y(x + a) + T_y(x) \neq 0, \forall a \neq 0, x \in \mathbb{F}_2^t.$$

Again $T_y(x + a) + T_y(x) \neq 0, \forall a \neq 0, x \in \mathbb{F}_2^t$ if and only if

$$F(x + a, y) + F(x, y) \neq (0, b), \forall a \neq 0, x \in \mathbb{F}_2^t \text{ and } b \in \mathbb{F}_2^{n-t}$$

as we have $F(x, y) = (T_y(x), U_x(y))$.

Now from the definition of DDT, $F(x + a, y) + F(x, y) \neq (0, b)$ is equivalent to $\delta_F(a, 0, 0, b) = 0, \forall (a, b) \neq (0, 0)$. Thus we get that T_y is a permutation $\forall y \in \mathbb{F}_2^{n-t}$ if and only if $(a, 0, 0, b) \in \mathcal{Z}_F^D, \forall (a, b) \neq (0, 0)$.

Let us consider $(0, 0, x_1, x_2) \in \nu^\perp$. Now if we apply t -swap on this vector we get $M_t(0, 0, x_1, x_2) = (x_1, 0, 0, x_2)$. So $M_t(0, 0, x_1, x_2) \in \mathcal{Z}_F^D$ if and only if T_y is a permutation $\forall y \in \mathbb{F}_2^{n-t}$. Thus finally we can get T_y is permutation $\forall y \in \mathbb{F}_2^{n-t}$ if and only if $M_t(\nu^\perp) \subseteq \mathcal{Z}_F^D$. \square

A direct corollary of this follows from Proposition 6.

Corollary 4 T_y is permutation $\forall y$ if and only if

$$M_t(\nu) \subseteq \mathcal{Z}_F.$$

As a consequence of this we get the following lemma.

Lemma 12 A function $P : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is permutation if and only if

$$\nu^\perp \subseteq \mathcal{Z}_F.$$

Proof:

Let us consider a TU projection, $P(x, y) = (T_y(x), U_x(y))$ for some t . Now if $t = n$, $P(x, y) = T(x)$, here T will be independent of y . So we get the following

$$\begin{aligned} P(x, y) \text{ is permutation} &\Leftrightarrow T(x) \text{ is permutation} \\ &\Leftrightarrow M_n(\nu) \subseteq \mathcal{Z}_F, \text{ from the above Corollary} \\ &\Leftrightarrow \nu^\perp \subseteq \mathcal{Z}_F. \end{aligned}$$

\square

Now we have all the results to prove the main theorem of this chapter.

Theorem 8 *A function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is CCZ-equivalent to a permutation P if and only if there exist two n -dimensional vector spaces $V, W \subseteq \mathcal{Z}_F$ such that*

$$\text{span}(V \cup W) = \mathbb{F}_2^{2n}.$$

Proof:

\Rightarrow Let F be CCZ-equivalent to P due to the affine permutation \mathcal{A} i.e $\Gamma_P = \mathcal{A}(\Gamma_F)$, where $\mathcal{A}(x) = \mathcal{L}(x) + c$. Now we have $\mathcal{Z}_F = \mathcal{L}^T(\mathcal{Z}_P)$ by Lemma 11.

Also from Lemma 12, as P is a permutation, we have $\nu^\perp \subseteq \mathcal{Z}_P$ and so we get

$$\mathcal{L}^T(\nu^\perp) \subseteq \mathcal{L}^T(\mathcal{Z}_P) = \mathcal{Z}_F.$$

Again as \mathcal{A} is admissible then from Theorem 7, $L^T(\nu) \subseteq \mathcal{Z}_F$. So we will take $V = L^T(\nu^\perp)$ and $W = L^T(\nu)$. Thus $V, W \subseteq \mathcal{Z}_F$ and obviously $\text{span}(V \cup W) = \mathbb{F}_2^{2n}$.

\Leftarrow Let the condition hold with $V, W \subseteq \mathcal{Z}_F$ i.e. $\text{span}(V \cup W) = \mathbb{F}_2^{2n}$. Then a $2n \times 2n$ matrix M can be constructed using a basis of V and W such that

$$M(\nu) = V \text{ and } M(\nu^\perp) = W.$$

Suppose V has a basis $\{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$ and W has a basis $\{\beta_0, \beta_1, \dots, \beta_{n-1}\}$. Then we will take these bases as columns of M i.e

$$M = \begin{bmatrix} \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \beta_0 & \beta_1 & \cdots & \beta_{n-1} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \end{bmatrix}$$

Now we will construct a function P such that $\Gamma_P = M^T(\Gamma_F)$, which is easy if we have F and M .

Claim: P is a permutation (Then F is CCZ-equivalent to a permutation).

Proof: As $\Gamma_P = M^T(\Gamma_F)$ then from Lemma 11, $\mathcal{Z}_P = ((M^T)^T)^{-1}(\mathcal{Z}_F) = M^{-1}(\mathcal{Z}_F)$.

Now as $W \subseteq \mathcal{Z}_F$, so

$$\begin{aligned} M^{-1}(W) &\subseteq M^{-1}(\mathcal{Z}_F) \\ \implies \nu^\perp &\subseteq \mathcal{Z}_P \end{aligned}$$

which implies that P is a permutation. This completes the proof. \square

So the above theorem is giving a necessary and sufficient condition for a function to be CCZ-equivalent to a permutation and also the way to construct the required permutation. Also, to find the vector spaces V and W we can use the vector space extraction algorithm, given in Chapter 2.

Chapter 4

Quadratic Indicator Cube (QIC)

In this chapter we will define how to write a quadratic homogeneous function in a cubic structure. Also we will describe the criteria related to this cube that is necessary and sufficient for a function to be APN. Then there will be some ideas to change the cube of an APN function to get new APN functions. We start with basic definitions on quadratic homogeneous functions and the construction of the cube in Section 4.1. In Section 4.2 we will provide our main idea to modify the cube. We will present our experimental results on the modification in Section 4.4 and before that in Section 4.3 we will provide important mathematical tools for the analysis of our result, namely the ortho-derivative and ortho-indicator.

Notions and Definitions

We will consider $\{e_0, e_1, \dots, e_{n-1}\}$ as the standard basis of \mathbb{F}_2^n , where $e_i = (0, 0, \dots, 1, \dots, 0)$ (1 in the i th position). Also we will take QH_n as the set of all quadratic homogeneous functions.

Definition 30 *A function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is in QH_n if and only if its coordinates F_k can be written as*

$$F_k(x_0, x_1, \dots, x_{n-1}) = \sum_{0 \leq i < j < n} Q_{i,j}^k \times x_i x_j, \text{ for all } k \in \{0, 1, \dots, n-1\}$$

where each $Q_{i,j}^k \in \mathbb{F}_2$.

Here let us consider $Q_{i,j}^k = Q_{j,i}^k$ and $Q_{i,i}^k = 0, \forall i, j$. Now we can write $F_k(x) = (U_k \times x) \cdot x$, for all $k \in \{0, 1, \dots, n-1\}$ where U_k is an upper triangular matrix,

defined as

$$U_k = \begin{bmatrix} 0 & Q_{01}^k & Q_{02}^k & \cdots & Q_{0n-1}^k \\ 0 & 0 & Q_{12}^k & \cdots & Q_{1n-1}^k \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Definition 31 A Quadratic Indicator Cube (QIC), denoted by Q is an ordered set of matrices $\{Q_0, Q_1, \dots, Q_{n-1}\}$ where $Q_k = U_k^T + U_k, \forall k \in \{0, 1, \dots, n-1\}$. So we have:

$$Q_k = \begin{bmatrix} 0 & Q_{0,1}^k & Q_{0,2}^k & \cdots & Q_{0,n-1}^k \\ Q_{0,1}^k & 0 & Q_{1,2}^k & \cdots & Q_{1,n-1}^k \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Q_{0,n-1}^k & Q_{1,n-1}^k & \cdots & \cdots & 0 \end{bmatrix}.$$

QIC and Derivatives

Let us consider a function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ in QH_n . Then from the definition of derivatives [13], the derivative of any coordinate F_k with respect to a canonical basis vector e_i is

$$\Delta_{e_i} F_k(x) = F_k(x + e_i) + F_k(x) = \sum_{j=0}^{n-1} Q_{i,j}^k \times x_j, \forall i, k. \quad (4.1)$$

So we get the following lemma.

Lemma 13 Let $Q = \{Q_{i,j}^k\}$ be a QIC of F . Then

$$\Delta_{e_j}(\Delta_{e_i} F_k(x)) = \Delta_{e_i} F_k(e_j) = Q_{i,j}^k, \forall i, j, k$$

Proof:

$$\begin{aligned} \Delta_{e_j}(\Delta_{e_i} F_k(x)) &= \Delta_{e_j} \left(\sum_{l=0}^{n-1} Q_{i,l}^k \times x_l \right) \\ &= Q_{i,j}^k \end{aligned}$$

Also $\Delta_{e_i} F_k(e_j) = Q_{i,j}^k$ directly follows from Equation (4.1). \square

Matrix representation of the Derivatives

For any function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ in QH_n we can get,

$$\begin{aligned} \Delta_{e_i} F(x) &= e_0 \Delta_{e_i} F_0(x) + \dots + e_{n-1} \Delta_{e_i} F_{n-1}(x) \\ &= (\Delta_{e_i} F_0(x), \dots, \Delta_{e_i} F_{n-1}(x)) \\ &= \left(\sum_{j=0}^{n-1} Q_{i,j}^0 x_j, \dots, \sum_{j=0}^{n-1} Q_{i,j}^{n-1} x_j \right). \end{aligned}$$

Thus we can consider the following matrix representation for all $i \in \{0, 1, \dots, n-1\}$,

$$\Delta_{e_i} F = \begin{bmatrix} Q_{i,0}^0 & Q_{i,1}^0 & Q_{i,2}^0 & \cdots & Q_{i,n-1}^0 \\ Q_{i,0}^1 & Q_{i,1}^1 & Q_{i,2}^1 & \cdots & Q_{i,n-1}^1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Q_{i,0}^{n-1} & Q_{i,1}^{n-1} & Q_{i,2}^{n-1} & \cdots & Q_{i,n-1}^{n-1} \end{bmatrix}.$$

So if we look carefully at the QIC , we can visualize it as shown in Figure 4.1, where the orientation (a) in the figure shows the directions of i, j and k . Thus if we take the first component of the function as front face of the cube then the derivatives will be walls and floors of cube.

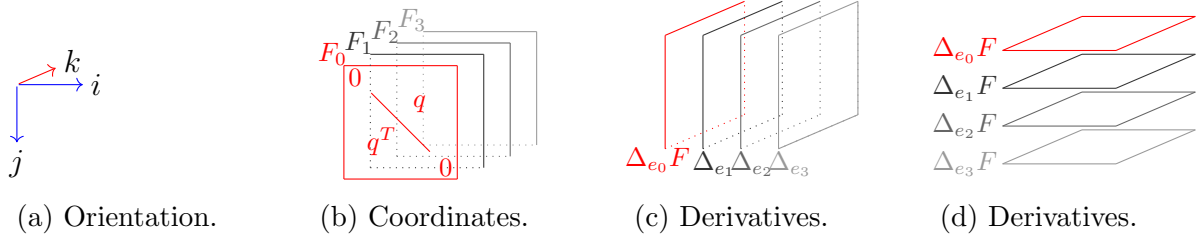


Figure 4.1: The meaning of the different parts of the QIC .

Lemma 14 Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a function. Then it holds that

$$\Delta_a F(x) = \sum_{i=0}^{n-1} a_i \Delta_{e_i} F(x + \sum_{j=0}^{i-1} a_j e_j)$$

for any vector $a = (a_0, a_1, \dots, a_{n-1})$ in \mathbb{F}_2^n .

Proof:

We will use induction on the Hamming weight of a . Suppose that $wt(a) = 1$ and let $a_i = 1$ for some $0 \leq i < n$, then $a = e_i$. So the right-hand side of the statement corresponds to $\Delta_{e_i} F(x + e_i) = \Delta_{e_i} F(x) = \Delta_a F(x)$. So the statement is true for the

base case. Suppose the lemma is true for all a with $wt(a) \leq m$. We have to show that the lemma is true for any a' such that $wt(a') = m + 1$.

Let a' be a vector with weight $m + 1$. Let us write $a' = a + e_k$ where $k = \max\{i : a'_i = 1\}$. So $wt(a) = m$. Then we get

$$\begin{aligned}
\Delta_{a'}F(x) &= F(x + a') + F(x) \\
&= F(x + a + e_k) + F(x + a) + F(x + a) + F(x) \\
&= \Delta_{e_k}F(x + a) + \Delta_aF(x) \\
&= \Delta_{e_k}F(x + a) + \sum_{i=0}^{n-1} a_i \Delta_{e_i}F(x + \sum_{j=0}^{i-1} a_j e_j), \text{ from the induction hypothesis} \\
&= \Delta_{e_k}F(x + a) + \sum_{i=0}^{k-1} a_i \Delta_{e_i}F(x + \sum_{j=0}^{i-1} a_j e_j) \\
&= \sum_{i=0}^k a'_i \Delta_{e_i}F(x + \sum_{j=0}^{i-1} a_j e_j) \\
&= \sum_{i=0}^{n-1} a'_i \Delta_{e_i}F(x + \sum_{j=0}^{i-1} a_j e_j).
\end{aligned}$$

So the statement is true for a' . Thus the lemma is true for any a . □

Theorem 9 *Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a quadratic homogeneous function. Then*

$$\Delta_a F(x) = \sum_{i=0}^{n-1} a_i \Delta_{e_i} F(x) + F(a).$$

Proof:

From the above lemma we have,

$$\Delta_a F(x) = \sum_{i=0}^{n-1} a_i \Delta_{e_i} F(x + \sum_{j=0}^{i-1} a_j e_j).$$

Now as $F \in QH_n$, all its derivatives along the canonical basis vectors e_i are linear.

Then

$$\begin{aligned}
\Delta_a F(x) &= \sum_{i=0}^{n-1} a_i \Delta_{e_i} F(x + \sum_{j=0}^{i-1} a_j e_j) \\
&= \sum_{i=0}^{n-1} a_i \Delta_{e_i} F(x) + \sum_{i=0}^{n-1} a_i \Delta_{e_i} F\left(\sum_{j=0}^{i-1} a_j e_j\right) \\
&= \left(\sum_{i=0}^{n-1} a_i \Delta_{e_i} F(x)\right) + \sum_{k=0}^{n-1} e_k \sum_{i=0}^{n-1} a_i \Delta_{e_i} F_k\left(\sum_{j=0}^{i-1} a_j e_j\right) \\
&= \left(\sum_{i=0}^{n-1} a_i \Delta_{e_i} F(x)\right) + \sum_{k=0}^{n-1} e_k \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} a_i a_j Q_{i,j}^k \\
&= \left(\sum_{i=0}^{n-1} a_i \Delta_{e_i} F(x)\right) + F(a).
\end{aligned}$$

□

This is an important theorem which will allow us to define an essential criterion on the QIC of a function to be APN.

The QIC of APN functions

Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a function. Then its derivative with respect to any vector $a \in \mathbb{F}_2^n$ can also be regarded as a function from \mathbb{F}_2^n to itself. The image set of this function is $\text{Im}(\Delta_a F) = \{F(x + a) + F(x) : x \in \mathbb{F}_2^n\}$ for any $a \in \mathbb{F}_2^n$. If F is an APN function then for any nonzero a , the cardinality of this set is 2^{n-1} . Now from the definition of QH_n , if F is in QH_n then the function $\Delta_a F + F(a) : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ defined as $(\Delta_a F + F(a))(x) = \Delta_a F(x) + F(a)$ is a linear map and so $\text{Im}(\Delta_a F)$ is an affine subspace.

Now considering $\text{Im}(\Delta_a F)$ as an affine subspace, we get the following corollary of Theorem 9.

Corollary 5 *If $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is a quadratic homogeneous function then*

$$\text{Im}(\Delta_a F) = F(a) + \left\{ \sum_{i=0}^{n-1} a_i \begin{bmatrix} Q_{i,0}^0 & Q_{i,1}^0 & \cdots & Q_{i,n-1}^0 \\ \cdots & \cdots & \cdots & \cdots \\ Q_{i,0}^{n-1} & Q_{i,1}^{n-1} & \cdots & Q_{i,n-1}^{n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ \cdots \\ x_{n-1} \end{bmatrix}, x \in \mathbb{F}_2^n \right\}$$

and consequently,

$$\dim(\text{Im}(\Delta_a F)) = \text{rank} \left(\sum_{i=0}^{n-1} a_i \begin{bmatrix} Q_{i,0}^0 & Q_{i,1}^0 & \cdots & Q_{i,n-1}^0 \\ \cdots & \cdots & \cdots & \cdots \\ Q_{i,0}^{n-1} & Q_{i,1}^{n-1} & \cdots & Q_{i,n-1}^{n-1} \end{bmatrix} \right).$$

Proof:

From Theorem 9 we have,

$$\Delta_a F(x) = F(a) + \sum_{i=0}^{n-1} a_i \Delta_{e_i} F(x).$$

Now replacing $\Delta_{e_i} F$ with its matrix representation, we get

$$\Delta_a F(x) = F(a) + \sum_{i=0}^{n-1} a_i \begin{bmatrix} Q_{i,0}^0 & Q_{i,1}^0 & \cdots & Q_{i,n-1}^0 \\ \cdots & \cdots & \cdots & \cdots \\ Q_{i,0}^{n-1} & Q_{i,1}^{n-1} & \cdots & Q_{i,n-1}^{n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ \cdots \\ x_{n-1} \end{bmatrix}.$$

So the first part is done. Therefore,

$$|\text{Im}(\Delta_a F(x))| = \left| \text{Im} \left(\sum_{i=0}^{n-1} a_i \begin{bmatrix} Q_{i,0}^0 & Q_{i,1}^0 & \cdots & Q_{i,n-1}^0 \\ \cdots & \cdots & \cdots & \cdots \\ Q_{i,0}^{n-1} & Q_{i,1}^{n-1} & \cdots & Q_{i,n-1}^{n-1} \end{bmatrix} \right) \right|.$$

Therefore the second statement is also true. \square

We can use this corollary to characterize the QIC of an APN function.

Proposition 9 *Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a quadratic homogeneous function. Then F is APN if and only if its QIC is such that*

$$\forall a \in \mathbb{F}_2^n \setminus \{0\}, \text{rank} \left(\sum_{i=0}^{n-1} a_i \begin{bmatrix} Q_{i,0}^0 & Q_{i,1}^0 & \cdots & Q_{i,n-1}^0 \\ \cdots & \cdots & \cdots & \cdots \\ Q_{i,0}^{n-1} & Q_{i,1}^{n-1} & \cdots & Q_{i,n-1}^{n-1} \end{bmatrix} \right) = n - 1.$$

Proof:

Let us recall the definition of APN function. A function F is APN if and only if for all $a \in \mathbb{F}_2^n \setminus \{0\}$ and $b \in \mathbb{F}_2^n$, $\#\{x \in \mathbb{F}_2^n : F(x+a) + F(x) = b\} \leq 2$. That is a function F is APN if and only if its derivative with respect to any nonzero vector is a 2-to-1 function. Which also means that a function $F \in QH_n$ is APN if and only if $\forall a, |\text{Im}(\Delta_a F)| = 2^{n-1}$. So by Corollary 5,

$$\text{rank} \left(\sum_{i=0}^{n-1} a_i \begin{bmatrix} Q_{i,0}^0 & Q_{i,1}^0 & \cdots & Q_{i,n-1}^0 \\ \cdots & \cdots & \cdots & \cdots \\ Q_{i,0}^{n-1} & Q_{i,1}^{n-1} & \cdots & Q_{i,n-1}^{n-1} \end{bmatrix} \right) = n - 1.$$

\square

Effect of Composition

Let $Q = \{Q_{i,j}^k\}$ be the QIC of the function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and let L be a linear function from \mathbb{F}_2^n to itself. Also, let $L = [L_{k,l}]$ be the matrix corresponding to L . Here we are interested to find the QIC corresponding to the function $L \circ F$. Let its QIC be $Q' = \{Q_{i,j}^k\}$. So for any k we get,

$$\begin{aligned} (L \circ F)_k(x_0, x_1, \dots, x_{n-1}) &= \sum_{l=0}^{n-1} L_{k,l} F_l(x_0, x_1, \dots, x_{n-1}) \\ &= \sum_{l=0}^{n-1} L_{k,l} \sum_{0 \leq i < j < n} Q_{i,j}^l x_i x_j \\ &= \sum_{0 \leq i < j < n} \left(\sum_{l=0}^{n-1} L_{k,l} Q_{i,j}^l \right) x_i x_j. \end{aligned}$$

Thus, we can write $Q_{i,j}^k = \sum_{l=0}^{n-1} L_{k,l} Q_{i,j}^l$. Again we will try to find the effect of composition on derivatives. From the construction of the derivative matrix we have,

$$\begin{aligned} \begin{bmatrix} Q_{i,0}^0 & Q_{i,1}^0 & \cdots & Q_{i,n-1}^0 \\ Q_{i,0}^1 & Q_{i,1}^1 & \cdots & Q_{i,n-1}^1 \\ \vdots & \vdots & \vdots & \vdots \\ Q_{i,0}^{n-1} & Q_{i,1}^{n-1} & \cdots & Q_{i,n-1}^{n-1} \end{bmatrix} &= \begin{bmatrix} \sum_{l=0}^{n-1} L_{0,l} Q_{i,0}^l & \sum_{l=0}^{n-1} L_{0,l} Q_{i,1}^l & \cdots & \sum_{l=0}^{n-1} L_{0,l} Q_{i,n-1}^l \\ \sum_{l=0}^{n-1} L_{1,l} Q_{i,0}^l & \sum_{l=0}^{n-1} L_{1,l} Q_{i,1}^l & \cdots & \sum_{l=0}^{n-1} L_{1,l} Q_{i,n-1}^l \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{l=0}^{n-1} L_{n-1,l} Q_{i,0}^l & \sum_{l=0}^{n-1} L_{n-1,l} Q_{i,1}^l & \cdots & \sum_{l=0}^{n-1} L_{n-1,l} Q_{i,n-1}^l \end{bmatrix} \\ &= \begin{bmatrix} L_{0,0} & L_{0,1} & \cdots & L_{0,n-1} \\ L_{1,0} & L_{1,1} & \cdots & L_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ L_{n-1,0} & L_{n-1,1} & \cdots & L_{n-1,n-1} \end{bmatrix} \times \begin{bmatrix} Q_{i,0}^0 & Q_{i,1}^0 & \cdots & Q_{i,n-1}^0 \\ Q_{i,0}^1 & Q_{i,1}^1 & \cdots & Q_{i,n-1}^1 \\ \vdots & \vdots & \vdots & \vdots \\ Q_{i,0}^{n-1} & Q_{i,1}^{n-1} & \cdots & Q_{i,n-1}^{n-1} \end{bmatrix} \\ &= L \times \Delta_{e_i} F. \end{aligned}$$

If L is a permutation then F and $L \circ F$ are EA-equivalent from Definition 22. So in that case if F is APN function then from Proposition 5 we can say $L \circ F$ is also APN. Also, this can be concluded from the previous discussion on the effect of composition on derivatives. Here, we have $rank(L \times \Delta_{e_i} F) = rank(\Delta_{e_i} F)$ and $rank(a_0 \{L \times \Delta_{e_0} F\} + a_1 \{L \times \Delta_{e_1} F\} + \dots + a_{n-1} \{L \times \Delta_{e_{n-1}} F\}) = rank(a_0 \times \Delta_{e_0} F + a_1 \times \Delta_{e_1} F + \dots + a_{n-1} \times \Delta_{e_{n-1}} F) \forall a = (a_0, a_1, \dots, a_{n-1}) \in \mathbb{F}_2^n \setminus \{0\}$ as L is a permutation. So if F is APN then $L \circ F$ is also APN from Proposition 9. Also, we can compose L

on the right of F . Let us consider $F' = F \circ L$. So we have

$$\begin{aligned}
F'_k(x_0, x_1, \dots, x_{n-1}) &= \sum_{0 \leq i < j < n} Q_{i,j}^k L(x)_i L(x)_j \\
&= \sum_{0 \leq i < j < n} Q_{i,j}^k \left(\sum_{u=0}^{n-1} L_{i,u} x_u \right) \left(\sum_{v=0}^{n-1} L_{j,v} x_v \right) \\
&= \sum_{0 \leq i < j < n} Q_{i,j}^k \left(\sum_{u=0}^{n-1} \sum_{v=0}^{n-1} L_{i,u} L_{j,v} x_u x_v \right) \\
&= \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \left(\sum_{0 \leq i < j < n} Q_{i,j}^k L_{i,u} L_{j,v} \right) x_u x_v.
\end{aligned}$$

Now the coefficient of $x_u x_v$ in the above sum is $\sum_{0 \leq i < j < n} Q_{i,j}^k L_{i,u} L_{j,v} + \sum_{0 \leq i < j < n} Q_{i,j}^k L_{i,v} L_{j,u}$ as the contribution to both $x_u x_v$ and $x_v x_u$. So we can write,

$$Q'_{u,v} = \sum_{0 \leq i < j < n} \left(Q_{i,j}^k L_{i,u} L_{j,v} + Q_{i,j}^k L_{i,v} L_{j,u} \right).$$

Thus, if we consider L_u as the u th row of the matrix corresponding to L and U_k as the upper triangular matrix,

$$U_k = \begin{bmatrix} 0 & Q_{0,1}^k & Q_{0,2}^k & \cdots & Q_{0,n-1}^k \\ 0 & 0 & Q_{1,2}^k & \cdots & Q_{1,n-1}^k \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

then we can write,

$$\begin{aligned}
Q'_{u,v} &= \sum_{0 \leq i < j < n} \left(Q_{i,j}^k L_{i,u} L_{j,v} + Q_{i,j}^k L_{i,v} L_{j,u} \right) \\
&= L_u U_k L_v^T + L_v U_k L_u^T \\
&= L_u U_k L_v^T + (L_v U_k L_u^T)^T \quad [\text{as } L_v U_k L_u^T \text{ is a scalar quantity}] \\
&= L_u U_k L_v^T + L_u U_k^T L_v^T \\
&= L_u U_k + U_k L_v^T \\
&= L_u Q_k L_v^T.
\end{aligned}$$

Therefore, we get $Q'^k = L Q^k L^T$.

Modifying the QIC

We will modify the QIC of an APN function in order to get a new APN function. By Proposition 9, after modification it will remain APN if we modify it in such a way that after modification any linear combination of the binary matrices corresponding to the derivatives with respect to canonical e_i , i.e $\Delta_{e_i}F$, has rank $n - 1$.

Modifying a Coordinate

Suppose F is a function in QH_n with QIC $Q = \{Q_{i,j}^k\}$. For a fixed k we get a binary symmetric matrix, say Q_k , where

$$Q_k = \begin{bmatrix} 0 & Q_{0,1}^k & Q_{0,2}^k & \cdots & Q_{0,n-1}^k \\ Q_{0,1}^k & 0 & Q_{1,2}^k & \cdots & Q_{1,n-1}^k \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Q_{0,n-1}^k & Q_{1,n-1}^k & \cdots & \cdots & 0 \end{bmatrix}.$$

This matrix Q_k for any $k \in \{0, 1, \dots, n - 1\}$ is called a *coordinate* of the QIC. In Figure 4.2, the gray face is the first coordinate(Q_0). We will try to replace a coordinate

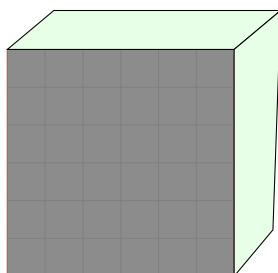


Figure 4.2: First coordinate(Q_0).

of the QIC of an APN function with a new binary symmetric matrix to obtain another QIC of an APN function. We will try to change the rows of the coordinate matrix one by one, preserving the rank property given in Proposition 9.

Algorithm

We can change any coordinate. Suppose we are changing the first coordinate i.e Q_0 . First we will prepare a list(L) corresponding to each row of the coordinate, called *fit-list*. The *fit-list* L_j will contain all possible vectors $v = (v_0, v_1, \dots, v_{n-1}) \in \mathbb{F}_2^n$ such that if we replace the first row of the matrix corresponding to $\Delta_{e_j}F$, the matrix

will still have rank $n - 1$ i.e

$$\text{rank} \left(\begin{bmatrix} v_0 & v_1 & \cdots & v_{n-1} \\ Q_{0,j}^1 & Q_{1,j}^1 & \cdots & Q_{n-1,j}^1 \\ \cdots & \cdots & \cdots & \cdots \\ Q_{0,j}^{n-1} & Q_{1,j}^{n-1} & \cdots & Q_{n-1,j}^{n-1} \end{bmatrix} \right) = n - 1.$$

As the original function F is an APN function, the rank of the original $\Delta_{e_j}F$ was also $n - 1$. So the size of this list L_j is at least one.

Now we will use a tree search to find a suitable vector from each *fit-list* and will construct a complete matrix with the BuildQIC function. This function will return a set, named *Result*, which contains lists of vectors. Each list will contain n vectors. Let $Result = \{Z^0, Z^1, \dots, Z^m\}$ and say $Z^k = \{Z_0, Z_1, \dots, Z_{n-1}\}$. Then we can construct a matrix with vectors in Z^k . We will replace the front face of a given QIC with this new matrix to get a new QIC.

Here we will use the same notation for vectors in Z within the algorithm i.e. $Z_{i,j}$ is the j th coordinate of i th vector in Z . Here we define another cube, called derivative cube, denoted by D where D is the set of derivative matrices with respect to the canonical basis i.e. $D = \{\Delta_{e_0}F, \Delta_{e_1}F, \dots, \Delta_{e_{n-1}}F\}$. Again if we look carefully then D is actually the cube generated using the floors of the original cube Q . Finding D from Q is easy. The subroutine ModifyFirstCoordinate to find *fit-lists* is specified in Algorithm 4.2.1, and the function BuildQIC is given in Algorithm 4.2.2.

Algorithm 4.2.1: MODIFYFIRSTCOORDINATE(D)

```

for  $j \leftarrow 0$  to  $n - 1$ 
  do  $\left\{ \begin{array}{l} M \leftarrow D_j \\ \textbf{for each } v = \{v_0, v_1, \dots, v_{n-1}\} \in \mathbb{F}_2^n \setminus \{0\} \\ \textbf{do} \left\{ \begin{array}{l} \text{replace 0th row of } M \text{ by } v \\ r \leftarrow \text{rank}(M) \\ \textbf{if } r = (n - 1) \\ \textbf{then Append } v \text{ to fit-list } L_j \end{array} \right. \end{array} \right.$ 
  Round  $\leftarrow 0$ 
   $Z \leftarrow \{\}$ 
  Result  $\leftarrow$  BUILDQIC(fitlist,  $D$ , Round,  $Z$ )

```

Algorithm 4.2.2: BUILDQIC($fitlist, D, Round, Z$)

```

if  $Round = n$ 
  then  $Result \leftarrow Result \cup \{Z\}$ 

  else  $\left\{ \begin{array}{l} \textbf{for each } v \in L_{Round} \\ \textbf{do} \left\{ \begin{array}{l} r \leftarrow \text{VERIFICATION}(v, fitlist, D, Round, Z) \\ \textbf{if } r = True \\ \textbf{then} \left\{ \begin{array}{l} Z_{new} \leftarrow Z \cup \{v\} \\ Result \leftarrow Result \cup \text{BUILDQIC}(fitlist, D, Round + 1, Z_{new}) \end{array} \right. \end{array} \right. \end{array} \right.$ 

return ( $Result$ )

```

Algorithm 4.2.3: VERIFICATION($v, fitlist, D, Round, Z$)

```

if  $Round = 0$ 
  then return ( $True$ )

  else  $\left\{ \begin{array}{l} Pass \leftarrow True \\ \textbf{comment:} \text{ First check that } v \text{ is symmetric with already added vectors in } Z \text{ or not} \\ \textbf{for } l \leftarrow 0 \text{ to } Round - 1 \\ \textbf{do} \left\{ \begin{array}{l} \textbf{if } v_l \neq Z_{l, Round} \\ \textbf{then return} (False) \end{array} \right. \\ \textbf{for } j \leftarrow 0 \text{ to } Round - 1 \\ \textbf{do} \left\{ \begin{array}{l} M_j \leftarrow D_j \\ \text{Replace 0th row of } M_j \text{ with the vector } Z_j \end{array} \right. \\ M_{Round} \leftarrow D_{Round} \\ \text{Replace 0th row of } M_{Round} \text{ with vector } v \\ \textbf{for each } a = (a_0, a_1, \dots, a_{Round-1}, 1, 0, \dots, 0) \in \mathbb{F}_2^n \setminus \{0\} \\ \textbf{do} \left\{ \begin{array}{l} M \leftarrow \sum_{l=0}^R a_l M_l \\ r \leftarrow rank(M) \\ \textbf{if } r \neq (n - 1) \\ \textbf{then } Pass \leftarrow False \end{array} \right. \\ \textbf{return} (Pass) \end{array} \right.$ 

```

Also, we have described the Verification function in Algorithm 4.2.3. This function will check whether the vector v from *fit-list* is suitable to add next. So this function is the most important. There we have to determine the rank of a Boolean matrix a lot of times. We have used the efficient rank finding Algorithm 2.4.1 discussed in Chapter 2.

Modifying One Sub-diagonal

Here we will replace the sub-diagonal of QIC with a set of n vectors. The gray vectors in Figure 4.3 are sub-diagonal vectors. We want to change those vectors. One

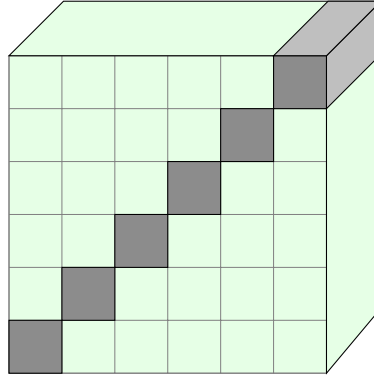


Figure 4.3: Sub-diagonal.

important observation here is that as the QIC is symmetric, we need to choose vectors only for the upper half of the sub-diagonal (see Figure 4.4).

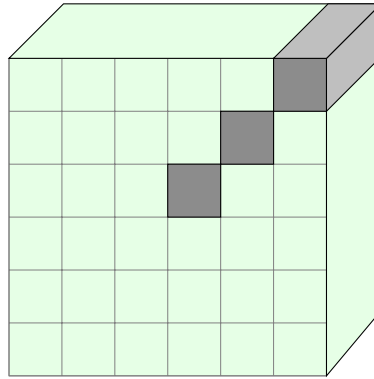


Figure 4.4: Upper half of the sub-diagonal

Algorithm

Again we will use the same process as we have used when changing one coordinate. First we will prepare a list, L , corresponding to each position in the upper half of the sub-diagonal, called *fit-list*. So here we will have $\frac{n-1}{2}$ *fit-lists*. The *fit-list* L_j will contain all possible vectors $v = (v_0, v_1, \dots, v_{n-1}) \in \mathbb{F}_2^n$ such that if we replace the $(n - 1 - j)$ th column of the matrix corresponding to $\Delta_{e_j} F$ (i.e the j th floor), the

matrix will still have rank $n - 1$ i.e

$$\text{rank} \left(\begin{bmatrix} Q_{0,j}^0 & Q_{1,j}^0 & \cdots & v_0 & \cdots & Q_{n-1,j}^0 \\ Q_{0,j}^1 & Q_{1,j}^1 & \cdots & v_1 & \cdots & Q_{n-1,j}^1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ Q_{0,j}^{n-1} & Q_{1,j}^{n-1} & \cdots & v_{n-1} & \cdots & Q_{n-1,j}^{n-1} \end{bmatrix} \right) = n - 1.$$

Moreover, observe that the $(n - 1 - j)$ th column of the matrix corresponding to $\Delta_{e_j}F$ is also the j th column of the matrix corresponding to $\Delta_{e_{n-1-j}}F$ due to symmetry. So we have a way to use a stronger filter to prepare a *fit-list* by checking the rank of this matrix. So with the above rank condition we need the following condition after replacing the j th column of the matrix corresponding to $\Delta_{e_{n-1-j}}F$.

$$\text{rank} \left(\begin{bmatrix} Q_{0,n-1-j}^0 & Q_{1,n-1-j}^0 & \cdots & v_0 & \cdots & Q_{n-1,n-1-j}^0 \\ Q_{0,n-1-j}^1 & Q_{1,n-1-j}^1 & \cdots & v_1 & \cdots & Q_{n-1,n-1-j}^1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ Q_{0,n-1-j}^{n-1} & Q_{1,n-1-j}^{n-1} & \cdots & v_{n-1} & \cdots & Q_{n-1,n-1-j}^{n-1} \end{bmatrix} \right) = n - 1.$$

This is shown in Figure 4.5. Two matrices are in yellow color and the common vector is gray.

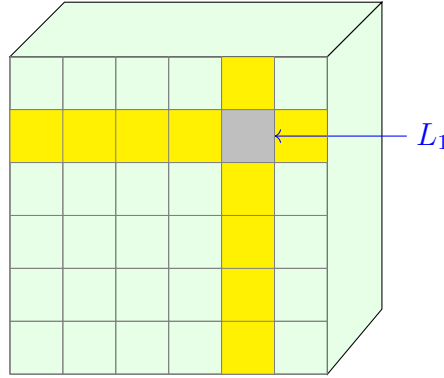


Figure 4.5: L_1 is the list of all vectors such that $\text{rank}(\Delta_{e_1}(F)) = 5$ and $\text{rank}(\Delta_{e_4}(F)) = 5$.

Then we will use a tree search to find suitable vectors from each *fit-list* and will construct a complete matrix with the BuildQIC function, which will return a set, named *Result*, which contains lists of vectors. But here each list will contain $\frac{n-1}{2}$ vectors. Let $\text{Result} = \{Z^0, Z^1, \dots, Z^m\}$ and say $Z^k = \{Z_0, Z_1, \dots, Z_{\frac{n-1}{2}}\}$. Then we will replace the upper half and the lower half of the sub-diagonal with these vectors. Obviously symmetric positions will be replaced by the same vector. The subroutine ModifyFirstCoordinate is specified in Algorithm 4.2.4, and the function BuildQIC is given in Algorithm 4.2.5.

Algorithm 4.2.4: MODIFYFIRSTCOORDINATE(D)

```

for  $j \leftarrow 0$  to  $\frac{n-1}{2}$ 
   $M_1 \leftarrow D_j$ 
   $M_2 \leftarrow D_{n-1-j}$ 
  for each  $v = \{v_0, v_1, \dots, v_{n-1}\} \in \mathbb{F}_2^n \setminus \{0\}$ 
    do  $\left\{ \begin{array}{l} \text{replace } (n-1-j)\text{th column of } M_1 \text{ by } v \\ \text{replace } j\text{th column of } M_2 \text{ by } v \end{array} \right.$ 
    do  $\left\{ \begin{array}{l} r_1 \leftarrow \text{rank}(M_1) \\ r_2 \leftarrow \text{rank}(M_2) \\ \text{if } r_1 = n-1 \text{ and } r_2 = n-1 \\ \text{then Append } v \text{ to fit-list } L_j \end{array} \right.$ 
   $Round \leftarrow 0$ 
   $Z \leftarrow \{\}$ 
   $Result \leftarrow \text{BUILDQIC}(fitlist, D, Round, Z)$ 

```

Algorithm 4.2.5: BUILDQIC($fitlist, D, Round, Z$)

```

if  $Round = \frac{n-1}{2}$ 
  then  $Result \leftarrow Result \cup \{Z\}$ 
else  $\left\{ \begin{array}{l} \text{for each } v \in L_{Round} \\ \text{do } \left\{ \begin{array}{l} r \leftarrow \text{VERIFICATION}(v, fitlist, D, Round, Z) \\ \text{if } r = True \\ \text{then } \left\{ \begin{array}{l} Z_{new} \leftarrow Z \cup \{v\} \\ Result \leftarrow Result \cup \text{BUILDQIC}(fitlist, D, Round + 1, Z_{new}) \end{array} \right. \end{array} \right. \end{array} \right.$ 
return ( $Result$ )

```

Now let us describe the verification algorithm. Here we will prepare two types of matrix, M_l and M_{l+1} from D_j and D_{n-1-j} . Then two symmetric positions will be replaced by the same vector from the list Z i.e the $(n-1-j)$ th column of M_l and the j th column of M_{l+1} will be replaced by the vector Z_j . Then we will check the rank of all possible linear combinations of those matrices. Note that at round $Round$, there will be a total of $2 \times Round + 2$ matrices. The function Verification is given in Algorithm 4.2.6.

Algorithm 4.2.6: VERIFICATION($v, \text{fitlist}, D, \text{Round}, Z$)

```

if  $\text{Round} = 0$ 
  then return ( $\text{True}$ )
   $\text{Pass} \leftarrow \text{True}$ 
   $l \leftarrow 0$ 
  for  $j \leftarrow 0$  to  $\text{Round} - 1$ 
     $M_l \leftarrow D_j$ 
    replace  $n - 1 - j$ th column of  $M_l$  by  $Z_j$ 
    do  $\left\{ \begin{array}{l} M_{l+1} \leftarrow D_{n-1-j} \\ \textit{replace } j\textit{th column of } M_{l+1} \textit{ by } Z_j \\ l \leftarrow l + 2 \end{array} \right.$ 
  else  $M_{2 \times \text{Round} + 1} \leftarrow D_{\text{Round}}$ 
  Replace  $(n - 1 - \text{Round})$ th column of  $M_{2 \times \text{Round} + 1}$  with vector  $v$ 
   $M_{2 \times \text{Round} + 2} \leftarrow D_{n-1-\text{Round}}$ 
  Replace  $\text{Round}$ -th column of  $M_{2 \times \text{Round} + 2}$  with vector  $v$ 
  for each  $a = (a_0, a_1, \dots, a_{2 \times \text{Round} + 2}, 0, 0, \dots, 0) \in \mathbb{F}_2^n \setminus \{0\}$ 
     $M \leftarrow \sum_{l=0}^R a_l M_l$ 
     $r \leftarrow \text{rank}(M)$ 
    if  $r \neq (n - 1)$ 
      then  $\text{Pass} \leftarrow \text{False}$ 
  return ( $\text{Pass}$ )

```

Modify Two Diagonals

Here we will describe the process to modify two diagonals together. This is the most important modification. Here two diagonals mean sub-diagonal and the diagonal lower than that. We can see the deep-gray sub-diagonal and light-gray diagonal lower than that in Figure 4.6. So we want to replace in total $2n - 2$ many vectors.

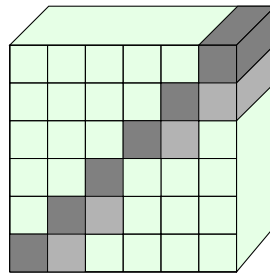


Figure 4.6: Two Diagonals

Here also, due to symmetry, choosing vectors for the upper part is enough. So we

have to choose a total of $\frac{n}{2} + \frac{n-2}{2} = n - 1$ many vectors as in Figure 4.7.

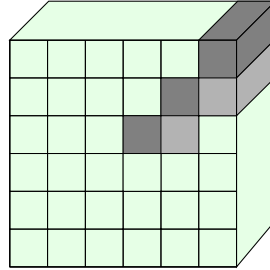


Figure 4.7: Upper Part

Algorithm

In this case we use a dynamic approach. There will be no pre-computation. We will pick vectors one by one for the positions from the upper-right-hand corner. We will pick one vector in each round. The order is given in Figure 4.8 for a QIC of size 6. Here we directly use the BuildQIC function, but without any pre-computation. Also

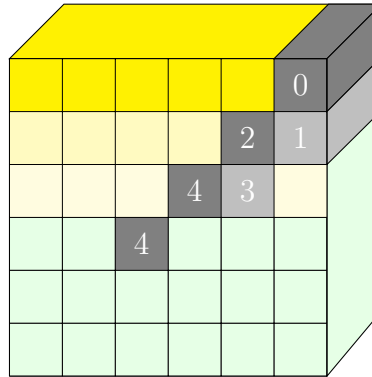


Figure 4.8: Order of picking vector

there is one function called SetVector, which will work for verification of a vector. If the vector is suitable for that position, it will set that vector into the derivative cube. Here both the position and the round are the same i.e at the r th round we choose a vector for the r th position. The function BuildQIC is given in Algorithm 4.2.7 and the function SetVector is given in Algorithm 4.2.9.

Algorithm 4.2.7: BUILDQIC($D, Round, Z$)

```

if  $Round = n - 1$ 
  then  $\left\{ \begin{array}{l} \text{Construct new QIC, } Q_{new} \text{ from } D \\ Result \leftarrow Result \cup \{Q_{new}\} \end{array} \right.$ 
  else  $\left\{ \begin{array}{l} \text{for each } v \in \mathbb{F}_2^n \setminus \{0\} \\ \quad \text{do } \left\{ \begin{array}{l} r \leftarrow \text{SETVECTOR}(v, D, Round) \\ \quad \text{if } r = True \\ \quad \quad \text{then } \left\{ \begin{array}{l} Z_{new} \leftarrow Z \cup \{v\} \\ Result \leftarrow Result \cup \text{BUILDQIC}(D, Round + 1) \end{array} \right. \end{array} \right.$ 
return ( $Result$ )

```

Now we will describe the SetVector function. Let us divide our algorithm in two parts. As we see in Figure 4.8, the verification of a vector depends on its position or round in QIC. So we will use two different approaches for even and odd positions.

Let us explain through an example. Suppose we have a QIC of size 6. Suppose we want to replace an even position, say position 2, where positions 0 and 1 are already set in both symmetric positions. Then to set a vector at position 2 we need to replace the $(6 - 2) = 4$ th column of the matrix corresponding to $\Delta_{e_1}F$ and the position symmetric to this. So we need to check the rank of all possible nonzero linear combinations of matrices corresponding to $\Delta_{e_0}F, \Delta_{e_1}F$ and $\Delta_{e_5}F$. See the yellow matrices in Figure 4.9.

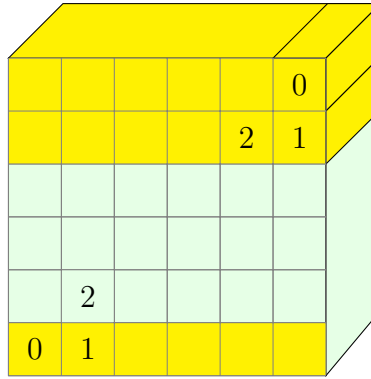


Figure 4.9: Setting vector at position 2.

Suppose now we need to change an odd position, say 3, where positions 0, 1 and 2 are already set. So here we replace the 4th column of the matrix corresponding to the matrix $\Delta_{e_2}F$ and the position symmetric to this. So here we need to check the rank of all possible nonzero linear combinations of matrix $\Delta_{e_0}F, \Delta_{e_5}F, \Delta_{e_1}F$ and $\Delta_{e_4}F$ (see the yellow matrices in figure 4.10).

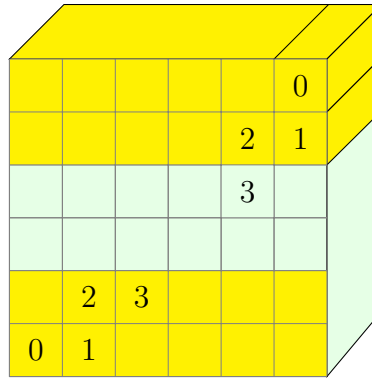
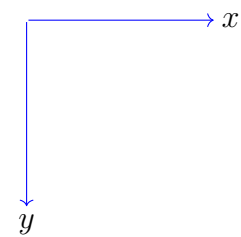


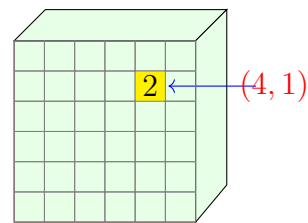
Figure 4.10: Setting vector at position 3.

So for the position m , if m is even we need to consider $m + 1$ matrices and if m is odd then also $m + 1$ matrices. Also, for the 4th position we need to consider one extra matrix, namely $\Delta_{e_3}F$. So the case of the last position needs to be handled separately.

Till now we have described using position (or round). But we need the coordinate corresponding to a position where we want to make the modification. Here we define a coordinate system where $(0, 0)$ means the vector at the upper-left-hand corner, see Figure 4.11a.



(a) Coordinate system



(b) Coordinate corresponding to round 2

For this we use a function called *Coordinate*, which will take the round number as input and outputs (x, y) . The function *Coordinate* is given in Algorithm 4.2.8.

Algorithm 4.2.8: COORDINATE(*Round*)

if (*Round is even*)
 then $\begin{cases} x = (n - 1) - \frac{Round}{2} \\ y = \frac{Round}{2} \end{cases}$
 else
 then $\begin{cases} x = (n - 1) - \frac{Round}{2} \\ y = \frac{Round}{2} + 1 \end{cases}$
return (*x, y*)

Algorithm 4.2.9: SETVECTOR($v, D, Round$)

$Pass \leftarrow True$

if $Round \neq (n - 2)$ **comment:** Will handle last round separately

then {

if $Round$ is even

$(x, y) \leftarrow \text{COORDINATE}(Round)$
 Replace x th column of D_y and y th column of D_x by v
 $l \leftarrow 0$
for $j \leftarrow 0$ **to** $y - 1$

do {

$M_l \leftarrow D_j$
 $M_{l+1} \leftarrow D_{n-1-j}$
 $l \leftarrow l + 2$

$M_{Round} \leftarrow D_y$
for each $a = (a_0, a_1, \dots, a_{Round}, 0, 0, \dots, 0) \in \mathbb{F}_2^n \setminus \{0\}$

do {

$M \leftarrow \sum_{l=0}^R a_l M_l$
 $r \leftarrow \text{rank}(M)$
if $r \neq (n - 1)$
then $Pass \leftarrow False$

else if $Round$ is odd

$(x, y) \leftarrow \text{COORDINATE}(Round)$
 Replace x th column of D_y and y th column of D_x by v
for $j \leftarrow 0$ **to** $y - 1$

do {

$M_l \leftarrow D_j$
 $M_{l+1} \leftarrow D_{n-1-j}$
 $l \leftarrow l + 2$

$a = (a_0, a_1, \dots, a_{Round}, 0, 0, \dots, 0) \in \mathbb{F}_2^n \setminus \{0\}$

do {

$M \leftarrow \sum_{l=0}^R a_l M_l$
 $r \leftarrow \text{rank}(M)$
if $r \neq (n - 1)$
then $Pass \leftarrow False$

else **comment:** Last round

then {

$(x, y) \leftarrow \text{COORDINATE}(Round)$
 Replace x th column of D_y and y th column of D_x by v
 $l \leftarrow 0$
for $j \leftarrow 0$ **to** y

do {

$M_l \leftarrow D_j$
 $M_{l+1} \leftarrow D_{n-1-j}$
 $l \leftarrow l + 2$

for each $a = (a_0, a_1, \dots, a_{Round}, 0, 0, \dots, 0) \in \mathbb{F}_2^n \setminus \{0\}$

do {

$M \leftarrow \sum_{l=0}^R a_l M_l$
 $r \leftarrow \text{rank}(M)$
if $r \neq (n - 1)$
then $Pass \leftarrow False$

return ($Pass$)

Ortho-Derivatives and Ortho-Indicators

We will present our experiment result in the next section. In this section we will present an important tool to check our result, called ortho-derivative. This will be useful to check if two functions are in different EA-equivalence classes.

Definition 32 *Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a quadratic function. Then we define the Ortho-Derivative of F as $\pi_F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and the Ortho-Indicator of F as $\sigma_F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ such that*

$$Im(\Delta_a F) = \{x \in \mathbb{F}_2^n : \pi_F(a) \cdot x = \sigma_F(a)\}.$$

We have already showed that if F is a quadratic function then $\forall a$, $\Delta_a F$ is an affine map. So the image sets of the derivatives of a quadratic function are always hyperplanes or complements of hyperplanes. So ortho-derivatives and ortho-indicators are well-defined.

Theorem 10 *Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and $G : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be two functions. If F and G are EA-equivalent then π_F and π_G are affine equivalent.*

Proof:

Let $G(x) = B \circ F \circ A(x) + C(x)$ where A and B are affine permutations with $A(x) = \mathcal{L}_A(x) + A'$ and $B(x) = \mathcal{L}_B(x) + B'$, where \mathcal{L}_A and \mathcal{L}_B are linear. Here note that without loss of generality we can take C as a linear permutation. Because if $C(x) = \mathcal{L}_C(x) + C'$ then we can add this C' to the constant part of B . So let C be simply a linear permutation. Now from the definition of the ortho-derivative for any a and x , we can write

$$\pi_G(a) \cdot \Delta_a G(x) = \sigma_G(a).$$

Thus we get

$$\begin{aligned} \pi_G(a) \cdot \Delta_a G(x) &= \sigma_G(a) \\ \Rightarrow \pi_G(a) \cdot [G(x+a) + G(x)] &= \sigma_G(a). \end{aligned}$$

Again from the relation between G and F we can write

$$\Rightarrow \pi_G(a) \cdot [B \circ F \circ A(x+a) + C(x+a) + B \circ F \circ A(x) + C(x)] = \sigma_G(a).$$

Now as C is a linear map we have:

$$\begin{aligned} \pi_G(a) \cdot [B \circ F \circ A(x+a) + C(x) + C(a) + B \circ F \circ A(x) + C(x)] &= \sigma_G(a) \\ \Rightarrow \pi_G(a) \cdot [B \circ F \circ A(x+a) + B \circ F \circ A(x)] &= \sigma_G(a) + \pi_G(a) \cdot C(a) \\ \Rightarrow \pi_G(a) \cdot [\mathcal{L}_B(F(A(x+a))) + B' + \mathcal{L}_B(F(A(x))) + B'] &= \sigma_G(a) + \pi_G(a) \cdot C(a) \\ \Rightarrow \pi_G(a) \cdot [\mathcal{L}_B(F(A(x+a))) + \mathcal{L}_B(F(A(x)))] &= \sigma_G(a) + \pi_G(a) \cdot C(a). \end{aligned}$$

Also we can deduce that

$$\begin{aligned} A(x+a) &= \mathcal{L}_A(x+a) + A' \\ &= \mathcal{L}_A(x) + \mathcal{L}_A(a) + A', \text{ as } \mathcal{L}_A \text{ is linear} \\ &= A(x) + \mathcal{L}_A(a). \end{aligned}$$

Thus from the above equation we get

$$\pi_G(a) \cdot [\mathcal{L}_B(F(A(x) + \mathcal{L}_A(a))) + \mathcal{L}_B(F(A(x)))] = \sigma_G(a) + \pi_G(a) \cdot C(a).$$

Here let us recall that for any vector v and any linear map L , we can get $v \cdot L(x) = (L^T \times v) \cdot x$. From this we can write the above equation as

$$\begin{aligned} &(\mathcal{L}_B^T \times \pi_G(a)) \cdot (F(A(x) + \mathcal{L}_A(a)) + F(A(x))) = \sigma_G(a) + \pi_G(a) \cdot C(a) \\ \Rightarrow &(\mathcal{L}_B^T \times \pi_G(a)) \cdot (F(y + \mathcal{L}_A(a)) + F(y)) = \sigma_G(a) + \pi_G(a) \cdot C(a), \text{ where } y = A(x) \\ \Rightarrow &(\mathcal{L}_B^T \times \pi_G(a)) \cdot \Delta_{\mathcal{L}_A(a)} F(y) = \sigma_G(a) + \pi_G(a) \cdot C(a). \end{aligned}$$

As the last equation is true for any $y \in \mathbb{F}_2^n$ we can write

$$\begin{aligned} \pi_F(\mathcal{L}_A(a)) &= (\mathcal{L}_B^T \times \pi_G(a)), \\ \sigma_F(a) &= \sigma_G(a) + \pi_G \cdot C(a). \end{aligned}$$

So $\pi_G(a) = (\mathcal{L}_B^T)^{-1} \circ \pi_F \circ \mathcal{L}_A(a)$ and then π_F and π_G are affine equivalent. This completes the proof. \square

The above theorem can be used as follows: If we have two functions F and G and if the corresponding π_F and π_G are not affine equivalent we can say F and G are in different EA-classes. To check whether π_F and π_G are in the same EA-equivalence class or not, we use Proposition 5 of Chapter 3. But for that we will use the following definition.

Definition 33 *Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a function. Then the i th differential number \mathcal{DS}_i is the total number of elements in the DDT with value i i.e.*

$$\mathcal{DS}_i = \#\{(a, b) \in \mathbb{F}_2^n \times \mathbb{F}_2^n : \delta_F(a, b) = i\}$$

and the i th linear number \mathcal{LS}_i is the total number of elements in the LAT with absolute value i i.e.

$$\mathcal{LS}_i = \#\{(a, b) \in \mathbb{F}_2^n \times \mathbb{F}_2^n : |\lambda_F(a, b)| = i\}.$$

Now we can construct a set of pairs called differential spectrum of F as

$$\mathcal{DS} = \{[i : \mathcal{DS}_i] : \mathcal{DS}_i \neq 0\}$$

and a set of pairs called linear spectrum as

$$\mathcal{LS} = \{[i : \mathcal{LS}_i] : \mathcal{LS}_i \neq 0\}.$$

Now from Proposition 5 of Chapter 3 if two functions π_F and π_G are affine equivalent (and then CCZ-equivalent) due to an EA-mapping \mathcal{A} with linear part \mathcal{L} then the relation between their DDT is

$$\delta_{\pi_G}(a, b) = \delta_{\pi_F}(\mathcal{L}^{-1}(a, b))$$

and the relation between their LAT is

$$\lambda_{\pi_G}(a, b) = (-1)^{c(a,b)} \lambda_{\pi_F}(\mathcal{L}^T(a, b))$$

So if π_F and π_G are affine equivalent then they will have the same differential spectrum (\mathcal{DS}) and the same linear spectrum (\mathcal{LS}).

Finally we can conclude the following lemma.

Lemma 15 *If π_F and π_G have different differential spectra or different linear spectra then F and G are in different EA-equivalence classes.*

Computing Ortho-Derivatives

To compute the ortho-derivative of a function efficiently we use the following lemma.

Lemma 16 *Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a quadratic function, then $\forall a \in \mathbb{F}_2^n$, we have*

$$\pi_F(a) \cdot (F(a) + F(0)) = \sigma_F(a).$$

Again, if $F \in QH_n$ then

$$\pi_F(a) \cdot F(a) = \sigma_F(a)$$

Proof:

From the definition of ortho-derivatives we have $\pi_F(a) \cdot \Delta_a F(x) = \sigma_F(a)$ and this holds for all $x \in \mathbb{F}_2^n$. So it must hold for $x = 0$. Thus

$$\begin{aligned} \pi_F(a) \cdot \Delta_a F(0) &= \sigma_F(a) \\ \implies \pi_F(a) \cdot (F(a) + F(0)) &= \sigma_F(a) \end{aligned}$$

Also if $F \in QH_n$ then $F(0) = 0$, so $\pi_F(a) \cdot F(a) = \sigma_F(a)$. □

Using the above lemma for any $F \in QH_n$ we can write

$$\begin{aligned} \pi_F(a) \cdot (F(x+a) + F(x)) &= \sigma_F(a) = \pi_F(a) \cdot F(a) \\ \implies \pi_F(a) \cdot (F(x+a) + F(x) + F(a)) &= 0 \end{aligned}$$

This result is used in Algorithm 4.3.1 to compute $\pi_F(a)$, which will return $\pi_F(a)$ as a look-up table.

Algorithm 4.3.1: ORTHODERIVATIVE(F)

```

for  $a \leftarrow 1$  to  $2^n - 1$ 
  do  $\left\{ \begin{array}{l} \text{for } R \leftarrow 1 \text{ to } 2^n - 1 \\ \text{do } \left\{ \begin{array}{l} Pass \leftarrow True \\ \text{for } x \leftarrow 0 \text{ to } 2^n - 1 \\ \text{do } \left\{ \begin{array}{l} \text{if } (R \cdot (F(x+a) + F(x) + F(a)) \neq 0) \\ \text{then } \left\{ \begin{array}{l} Pass \leftarrow False \\ \text{break} \end{array} \right. \\ \text{if } (Pass = True) \\ \text{then } \left\{ \begin{array}{l} \pi_F(a) \leftarrow R \\ \text{break} \end{array} \right. \end{array} \right. \end{array} \right. \end{array} \right.$ 
return  $(\pi_F)$ 

```

Experimental Results

Here we will provide results of the modification algorithms presented in Section 4.2. For example let us consider the function $F(x) = x^3$ from \mathbb{F}_{2^6} to itself. We have presented the algebraic normal form of this function in Chapter 2. Let us recall that again. Let us consider that the irreducible polynomial corresponding to the field \mathbb{F}_{2^6} is $x^6 + x^4 + x^3 + x + 1$. Then the algebraic normal forms of its 6 coordinates are as follows

- $x_0x_3 + x_0x_4 + x_0x_5 + x_0 + x_1x_5 + x_2x_3 + x_2 + x_3 + x_4x_5 + x_4$
- $x_0x_1 + x_0x_3 + x_0x_4 + x_1x_3 + x_1x_4 + x_1x_5 + x_2x_4 + x_2 + x_4x_5$
- $x_0x_1 + x_0x_2 + x_0x_4 + x_1x_3 + x_1x_4 + x_1x_5 + x_2x_4 + x_2x_5 + x_3 + x_5$
- $x_1x_4 + x_1x_5 + x_1 + x_2x_5 + x_2 + x_3x_4 + x_4x_5 + x_4$
- $x_0x_2 + x_0x_3 + x_0x_5 + x_1x_2 + x_1x_3 + x_2x_5 + x_2 + x_3x_5 + x_3 + x_4x_5$
- $x_0x_4 + x_1x_2 + x_1x_4 + x_3x_5 + x_3 + x_4 + x_5$.

The above ANFs have degree 2 and degree 1 terms. So this function is not in QH_6 . But we can simply remove the non-quadratic terms from each coordinate because removing degree-1 terms does not change the EA-equivalence class. So after removing degree-1 terms we get the following form

- $x_0x_3 + x_0x_4 + x_0x_5 + x_1x_5 + x_2x_3 + x_4x_5$
- $x_0x_1 + x_0x_3 + x_0x_4 + x_1x_3 + x_1x_4 + x_1x_5 + x_2x_4 + x_4x_5$

- $x_0x_1 + x_0x_2 + x_0x_4 + x_1x_3 + x_1x_4 + x_1x_5 + x_2x_4 + x_2x_5$
- $x_1x_4 + x_1x_5 + x_2x_5 + x_3x_4 + x_4x_5$
- $x_0x_2 + x_0x_3 + x_0x_5 + x_1x_2 + x_1x_3 + x_2x_5 + x_3x_5 + x_4x_5$
- $x_0x_4 + x_1x_2 + x_1x_4 + x_3x_5$.

Now from these ANF of coordinates we can find the QIC as our given construction. Let us describe it for the first coordinate. For the first coordinate we have

$$F_0(x_0, x_1, \dots, x_5) = x_0x_3 + x_0x_4 + x_0x_5 + x_1x_5 + x_2x_3 + x_4x_5.$$

So we can write

$$Q_0 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

It is well-known that this function $F(x) = x^3$ from \mathbb{F}_{2^6} to itself is an APN function. So we can modify the QIC of this function to get a new APN function using our modification algorithms. Our steps of experiment are as follows:

1. If the given function is not quadratic homogeneous, change it to quadratic homogeneous.
2. Find the QIC (and if required, Derivative QIC) from the quadratic homogeneous function.
3. Modify the QIC to get new functions and store the functions in a list.
4. Find ortho-derivative π_F for each new function.
5. Cluster the functions using differential spectra and linear spectra of the ortho-derivative π_F .

Recall the fact that if two functions are in the same EA-equivalence class then their ortho-derivatives will have the same differential spectra and linear spectra (Theorem 10). So the clustering in the last step will give different EA-equivalence classes. The clustering program is implemented in SageMath using Procedure 4.3.1 (to find ortho-derivative). The clustering program takes a list of functions as input and outputs a hash table. The keys of this hash table are pair of differential spectrum and linear spectrum and the values are list of functions such that their ortho-derivatives have the differential spectrum and linear spectrum specified by the key. We have implemented all of our algorithms in a machine with the configuration given in Appendix B.

Modifying a Coordinate

We have implemented this algorithm in SageMath. From the above function $F(x) = x^3$ in \mathbb{F}_{2^6} , we got a total of 32 new functions by modifying the first coordinate. The time required for this is 72s. The clustering program gives only one cluster with key i.e.

$$\begin{aligned} &(\text{differential spectrum, linear spectrum}) = \\ &(\{64 : 1, 0 : 2268, 2 : 1764, 8 : 63\}, \{32 : 1, 8 : 588, 4 : 1680, 0 : 1827\}). \end{aligned}$$

This result is interesting, so we investigate further. We want to figure out whether modifying a coordinate can change EA-equivalence class or not. Recall that two functions F and G are in the same EA-equivalence class if and only if we can find two affine permutations A and B and one affine function C such that $G(x) = B \circ F \circ A(x) + C(x)$. Now let us take A as the identity matrix and C as the Zero matrix. Then we get the equation $G(x) = B \circ F(x)$ where

$$F(x) = \begin{bmatrix} F_0(x) \\ F_1(x) \\ F_2(x) \\ \vdots \\ F_{n-1}(x) \end{bmatrix}.$$

Also by construction of the QIC we can write each F_k as $F_k(x) = (U_k \times x) \cdot x$ and

$$U_k = \begin{bmatrix} 0 & Q_{0,1}^k & Q_{0,2}^k & \cdots & Q_{0,n-1}^k \\ 0 & 0 & Q_{1,2}^k & \cdots & Q_{1,n-1}^k \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

So if we can find a matrix B with rank n (i.e B is a permutation) then we can say G is also in the same EA-equivalence class as F . Now G has the same coordinate as F except the 0th coordinate because we have changed only that coordinate. So we can take the matrix

$$B = \begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_{n-1} \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

We will get a function G which will be different from F but in the same EA-equivalence

class if $b_0 = 1$ and at least one of $b_i = 1$ for $i \in \{1, 2, \dots, n-1\}$. Let us take

$$G(x) = \begin{bmatrix} G_0(x) \\ F_1(x) \\ F_2(x) \\ \vdots \\ F_{n-1}(x) \end{bmatrix}.$$

Then from $G(x) = B \circ F(x)$ we get $G_0(x) = b_0 F_0(x) + b_1 F_1(x) + \dots + b_{n-1} F_{n-1}(x)$. Now each $F_k(x) = U_k(x) \cdot x$ and let $G_k(x) = U'_k(x) \cdot x$ where each U_k or U'_k has $\frac{n(n-1)}{2}$ elements as they are upper triangular matrices. Now suppose

$$U'_0 = \begin{bmatrix} 0 & c_1 & c_2 & \cdots & c_{n-1} \\ 0 & 0 & c_n & \cdots & c_{2n-3} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & c_{\frac{n(n-1)}{2}} \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

So from the relation $U'_0 = b_0 U_0 + b_1 U_1 + \dots + b_{n-1} U_{n-1}$ we can get a system of linear equations $H \times b = c$ where

$$b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix}$$

and

$$c = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{\frac{n(n-1)}{2}} \end{bmatrix}$$

and H is a $\frac{n(n-1)}{2} \times n$ order matrix. Now let us divide the matrix H in $n-1$ parts as

$$H = \begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ \vdots \\ H_{n-1} \end{bmatrix}$$

where H_j is of order $(n - 1 - j) \times n$.

Here H_j corresponds to j th row of the matrices $U_0, U_1 \cdots U_{n-1}$. So we get

$$H_j = \begin{bmatrix} Q_{j,j+1}^0 & Q_{j,j+1}^1 & Q_{j,j+1}^2 & \cdots & Q_{j,j+1}^{n-1} \\ Q_{j,j+2}^0 & Q_{j,j+2}^1 & Q_{j,j+2}^2 & \cdots & Q_{j,j+2}^{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Q_{j,n-1}^0 & Q_{j,n-1}^1 & Q_{j,n-1}^2 & \cdots & Q_{j,n-1}^{n-1} \end{bmatrix}$$

and as this is a QIC of an APN, all the rows of H_j are linearly independent. So the rank of H_j is $n - 1 - j$.

Now with this system of linear equations our observation is that when we are trying to solve this set of equations we are getting a unique solution for each set of equations of the form

$$y = \begin{bmatrix} 1 \\ b_1 \\ b_2 \\ \vdots \\ b_{\frac{n(n-1)}{2}} \end{bmatrix}.$$

So we get a matrix B with rank n such that $G(x) = B \circ F(x)$. This experiment seems to indicate that modifying one coordinate can not change the EA-equivalence class. However it is not the case as we can show by the following theorem from Budaghyan et al.'s work in [7].

Theorem 11 *Let F be a quadratic APN function from \mathbb{F}_{2^n} to itself, let f be a quadratic Boolean function on \mathbb{F}_{2^n} and*

$$\phi_F(x, a) = F(x) + F(x + a) + F(a) + F(0)$$

$$\phi_f(x, a) = f(x) + f(x + a) + f(a) + f(0).$$

If for every nonzero $a \in \mathbb{F}_{2^n}$ there exists a linear Boolean function l_a satisfying the conditions

$$\phi_f(x, a) = l_a(\phi_F(x, a)),$$

$$\text{If } \phi_F(x, a) = 1 \text{ for some } x \in \mathbb{F}_{2^n} \text{ then } l_a(1) = 0.$$

Then the function $F(x) + f(x)$ is also APN.

Here one important corollary of this theorem is that for any positive integer n , the function $x^3 + \text{tr}(x^9)$ is APN on \mathbb{F}_{2^n} . To prove this we will apply the above theorem with $F(x) = x^3$. Also, let $\phi_F(x, a) = a^2x + ax^2$, $f(x) = \text{tr}(x^9)$, $\phi_f(x, a) = \text{tr}(a^8x + ax^8)$ and $l_a(y) = \text{tr}(a^6y + a^3y^2 + a^{-3}y^4)$. Then we have,

$$\begin{aligned} l_a(\phi_F(x, a)) &= \text{tr}(a^6(a^2x + ax^2) + a^3(a^2x + ax^2)^2 + a^{-3}(a^2x + ax^2)^4) \\ &= \text{tr}(a^6(a^2x + ax^2) + a^3(a^4x^2 + a^2x^4)^2 + a^{-3}(a^8x^4 + a^4x^8)^4) \\ &= \phi_f(x, a) \end{aligned}$$

Also if for some $x \in \mathbb{F}_{2^n}$, $\phi_F(x, a) = 1$ then

$$\begin{aligned} l_a(1) &= \text{tr}(a^6 + a^3 + a^{-3}) \\ &= \text{tr}(a^{-3}) \\ &= \text{tr}\left(\frac{a^2x + ax^2}{a^3}\right) \\ &= \text{tr}\left(\frac{x}{a} + \left(\frac{x}{a}\right)^2\right) \\ &= 0 \end{aligned}$$

Thus from Theorem 11 we can say $x^3 + \text{tr}(x^9)$ is an APN function. Also we state without proof from Budaghyan et al.'s work that x^3 and $x^3 + \text{tr}(x^9)$ are in different EA-equivalence classes.

As the tr function maps to \mathbb{F}_2 , so $\text{tr}(x^9)$ changes exactly one bit of x^3 . On the other hand changing the first coordinate with our modification algorithm also changes one bit. So one of the modified functions should be equal to the function $x^3 + \text{tr}(x^9)$. But we get only one EA-equivalence class in our result. It seems to be a contradiction. But the argument is that $\text{tr}(x^9)$ changes one coordinate, but not necessarily the first coordinate. So if we try the modification with all possible components of F i.e. $cF(x)$ for all $c \in \mathbb{F}_{2^n}^*$, we can get at least two EA-equivalence classes with our algorithm. In the paper [29], the authors mention that changing a single coordinate produces functions in the same EA-equivalence class. The above result prove that this does not hold in general. We have found two EA-equivalence classes by modifying $cF(x)$ for $c = 2, 3, 4$ and 5 . In each case we get total 64 functions, which are equally divided into two classes. These are the list of (differential spectrum, linear spectrum) or keys:

- $(\{64 : 1, 0 : 2646, 2 : 1008, 4 : 378, 8 : 63\}, \{32 : 1, 8 : 462, 4 : 1617, 0 : 1953, 12 : 63\})$
- $(\{64 : 1, 0 : 2268, 2 : 1764, 8 : 63\}, \{32 : 1, 8 : 588, 4 : 1680, 0 : 1827\})$.

Modifying One Sub-Diagonal

We have implemented this algorithm in SageMath. From the above function $F(x) = x^3$ in \mathbb{F}_{2^6} , we got a total of 3 new functions by modifying one sub-diagonal. The time required for this is 7 seconds. The clustering program gives three clusters with the following list of (differential spectrum, linear spectrum) keys:

- $(\{64 : 1, 0 : 2477, 2 : 1271, 4 : 303, 6 : 37, 8 : 7\}, \{32 : 1, 0 : 832, 2 : 1427, 4 : 991, 6 : 549, 8 : 206, 10 : 71, 12 : 17, 14 : 1, 16 : 1\})$
- $(\{64 : 1, 0 : 2268, 2 : 1764, 8 : 63\}, \{32 : 1, 8 : 588, 4 : 1680, 0 : 1827\})$
- $(\{64 : 1, 0 : 2467, 2 : 1307, 4 : 261, 6 : 53, 8 : 7\}, \{32 : 1, 0 : 817, 2 : 1456, 4 : 997, 6 : 510, 8 : 222, 10 : 78, 12 : 11, 14 : 4\})$.

So we are getting three different classes for three new functions.

Modifying Two Diagonals

We have implemented this algorithm in C++. Here we got the most important results. From the above function $F(x) = x^3$ in \mathbb{F}_{2^6} , we got a total of 5036 new functions by modifying two diagonals. The time required for this is 64 seconds and 24s with 8-threads. The clustering program gives a total of 13 clusters. The (differential spectrum, linear spectrum) and the distribution of the 5036 functions are given in Table 4.1. Also, according to Edel's result [15], there is only 13 possible classes for \mathbb{F}_{2^6} . So our algorithm is giving all possible classes and the Kim mapping, which is CCZ-equivalent to Dillon's permutation, belongs to the second last class of the Table 4.1.

(Differential spectrum, linear spectrum)	#functions
$\{64 : 1, 0 : 2517, 2 : 1176, 4 : 370, 6 : 30, 10 : 2\},$ $\{32 : 1, 0 : 858, 2 : 1436, 4 : 953, 6 : 537, 8 : 220, 10 : 73, 12 : 15, 14 : 2, 16 : 1\}$	154
$\{64 : 1, 0 : 2489, 2 : 1255, 4 : 297, 6 : 49, 8 : 5\},$ $\{32 : 1, 0 : 802, 2 : 1439, 4 : 1031, 6 : 534, 8 : 196, 10 : 72, 12 : 17, 14 : 3, 16 : 1\}$	646
$\{64 : 1, 0 : 2467, 2 : 1307, 4 : 261, 6 : 53, 8 : 7\},$ $\{32 : 1, 0 : 817, 2 : 1456, 4 : 997, 6 : 510, 8 : 222, 10 : 78, 12 : 11, 14 : 4\}$	684
$\{64 : 1, 0 : 2477, 2 : 1271, 4 : 303, 6 : 37, 8 : 7\},$ $\{32 : 1, 0 : 832, 2 : 1427, 4 : 991, 6 : 549, 8 : 206, 10 : 71, 12 : 17, 14 : 1, 16 : 1\}$	681
$\{64 : 1, 0 : 2646, 2 : 1008, 4 : 378, 8 : 63\},$ $\{32 : 1, 8 : 462, 4 : 1617, 0 : 1953, 12 : 63\}$	10
$\{64 : 1, 0 : 2505, 2 : 1229, 4 : 303, 6 : 51, 8 : 7\},$ $\{32 : 1, 0 : 858, 2 : 1438, 4 : 958, 6 : 534, 8 : 212, 10 : 74, 12 : 18, 14 : 2, 16 : 1\}$	651
$\{64 : 1, 0 : 2401, 2 : 1428, 4 : 210, 6 : 56\},$ $\{32 : 1, 8 : 462, 4 : 1995, 0 : 1617, 12 : 21\}$	92
$\{64 : 1, 0 : 2485, 2 : 1271, 4 : 279, 6 : 53, 8 : 7\},$ $\{32 : 1, 0 : 854, 2 : 1434, 4 : 960, 6 : 540, 8 : 216, 10 : 72, 12 : 16, 14 : 2, 16 : 1\}$	644
$\{64 : 1, 0 : 2464, 2 : 1371, 4 : 195, 6 : 50, 14 : 15\},$ $\{32 : 1, 0 : 933, 2 : 1486, 4 : 848, 6 : 468, 8 : 260, 10 : 88, 14 : 6, 16 : 6\}$	76
$\{64 : 1, 0 : 2502, 2 : 1235, 4 : 297, 6 : 57, 8 : 4\},$ $\{32 : 1, 0 : 824, 2 : 1427, 4 : 1010, 6 : 552, 8 : 190, 10 : 66, 12 : 22, 14 : 3, 16 : 1\}$	620
$\{64 : 1, 0 : 2448, 2 : 1339, 4 : 258, 6 : 45, 8 : 2, 12 : 3\},$ $\{32 : 1, 0 : 821, 2 : 1440, 4 : 1000, 6 : 531, 8 : 210, 10 : 75, 12 : 16, 14 : 2\}$	731
$\{64 : 1, 0 : 2436, 2 : 1428, 4 : 168, 8 : 63\},$ $\{32 : 1, 8 : 364, 4 : 2072, 0 : 1603, 12 : 56\}^*$	44
$\{64 : 1, 0 : 2268, 2 : 1764, 8 : 63\},$ $\{32 : 1, 8 : 588, 4 : 1680, 0 : 1827\}$	3

Table 4.1: Result of the modification of 2 diagonals.

*The Kim mapping belongs to this class

(Function)	<i>#thread</i> = 1	<i>#thread</i> = 16
A.0.1	16709s	2489s
A.0.2	15556s	2901s
A.0.3	15524s	1981s
A.0.4	15388s	2120s
A.0.5	14490s	3278s
A.0.6	16710s	1783s

Table 4.2: Time required for the modification of 2 diagonals of QIC of size 8.

We have also applied our modification algorithms to the QIC of some functions over \mathbb{F}_2^8 . We have applied our algorithms on the APN functions given in [29]. The quadratic homogeneous form of some examples from [29] are given in the Appendix as a look-up table. So far we have not found any new APN function for the example we have tried. But our modification algorithms run successfully and the required time is given in Table 4.2 with reference to the Appendix. So it seems that it is not possible to find new APN functions just by changing 2 diagonals for 8-bit.

Conclusion

In this work, we presented a new way to tackle the ‘Big APN problem’. In particular, we have presented an important mathematical concept regarding quadratic homogeneous function by proposing the idea of representing a quadratic vectorial Boolean function using a 3-dimensional cube. Also, we have found a criterion related to this cube that is necessary and sufficient for a function to be APN. Then we have presented some algorithms based on backtracking to change the elements of the cube in such a way that if we start from an APN function it remains APN. For 6-variable APN function our algorithms have found many new functions that are EA-equivalent to all known classes. We have provided our results and the classification of new results in this thesis. For 8-bit APN functions our algorithm is running successfully, but so far we have not found any new function. Thus, our results for 8-variable indicate that the density of 8-variable APN function in the set of quadratic vectorial Boolean functions is low.

Future Work

One of the most important questions is the existence of 8-variable APN permutation. Our algorithms are slow when the number of variables is 8 or more. With this idea of 3-dimensional cube, one direction can be finding a new efficient algorithm for modification. Another direction can be the use of this 3-dimensional cube to study various properties of quadratic homogeneous functions.

Bibliography

- [1] Adhikari, M., Adhikari, A.: Basic Modern Algebra with Applications (01 2014)
- [2] Berger, T., Canteaut, A., Charpin, P., Laigle-Chapuy, Y.: On almost perfect nonlinear functions over \mathbb{F}_2^n . *Information Theory, IEEE Transactions on* 52, 4160 – 4170 (10 2006)
- [3] Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO'90. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (Aug 1991)
- [4] Bonnetain, X., Perrin, L., Tian, S.: Anomalies and vector space search: Tools for S-box analysis. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 196–223. Springer, Heidelberg (Dec 2019)
- [5] Brinkmann, M., Leander, G.: On the classification of APN functions up to dimension five. *Des. Codes Cryptography* 49(1–3), 273–288 (Dec 2008), <https://doi.org/10.1007/s10623-008-9194-6>
- [6] Browning, K., Dillon, J., McQuistan, M., Wolfe, A., McGuire, G.: An apn permutation in dimension six, 9th, international conference on finite fields and applications; finite fields: theory and applications /. In: CONTEMPORARY MATHEMATICS, International conference on finite fields and applications; Finite fields: theory and applications /, 9th, International conference on finite fields and applications; Finite fields: theory and applications /. pp. 33–42. No. 518, Eurospan [distributor], Providence, R.I. (2010), <https://www.tib.eu/de/suchen/id/BLCP%3ACN077150090>
- [7] Budaghyan, L., Carlet, C., Leander, G.: Constructing new APN functions from known ones. *Finite Fields and Their Applications* 15(2), 150 – 159 (2009), <http://www.sciencedirect.com/science/article/pii/S1071579708000622>
- [8] Canteaut, A.: Lecture Notes on Cryptographic Boolean Functions. <https://www.rocq.inria.fr/secret/Anne.Canteaut/>

- [9] Canteaut, A., Perrin, L.: On CCZ-equivalence, extended-affine equivalence, and function twisting. *Finite Fields and Their Applications* 56, 209 – 246 (2019), <http://www.sciencedirect.com/science/article/pii/S1071579718301485>
- [10] Carlet, C., Charpin, P., Zinoviev, V.: Codes, Bent functions and permutations suitable for DES-like cryptosystems. *Des. Codes Cryptography* 15, 125–156 (11 1998)
- [11] Chabaud, F., Vaudenay, S.: Links between differential and linear cryptanalysis. In: De Santis, A. (ed.) *Advances in Cryptology — EUROCRYPT'94*. pp. 356–365. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
- [12] Dobbertin, H.: Almost perfect nonlinear power functions on $\text{GF}(2^n)$: the Welch case. *IEEE Transactions on Information Theory* 45(4), 1271–1275 (1999)
- [13] Dobbertin, H.: Almost perfect nonlinear power functions on $\text{GF}(2^n)$: The Niho case. *Information and Computation* 151(1), 57 – 72 (1999), <http://www.sciencedirect.com/science/article/pii/S089054019892764X>
- [14] Dobbertin, H.: Almost perfect nonlinear power functions on $\text{GF}(2^n)$: A new case for n divisible by 5. In: Jungnickel, D., Niederreiter, H. (eds.) *Finite Fields and Applications*. pp. 113–121. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
- [15] Edel, Y.: Quadratic apn functions as subspaces of alternating bilinear forms (2010)
- [16] Gold, R.: Maximal recursive sequences with 3-valued recursive cross-correlation functions (corresp.). *IEEE Transactions on Information Theory* 14(1), 154–156 (1968)
- [17] Heys, H.: A tutorial on linear and differential cryptanalysis. *Cryptologia* 26 (06 2001)
- [18] Janwa, H., Wilson, R.M.: Hyperplane sections of fermat varieties in P^3 in char. 2 and some applications to cyclic codes. In: Cohen, G., Mora, T., Moreno, O. (eds.) *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*. pp. 180–194. Springer Berlin Heidelberg, Berlin, Heidelberg (1993)
- [19] Katz, J., Lindell, Y.: *Introduction to Modern Cryptography* (Chapman and Hall/Crc Cryptography and Network Security Series). Chapman and Hall/CRC (2007)
- [20] Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseht, T. (ed.) *EUROCRYPT'93*. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (May 1994)

- [21] Nyberg, K.: Perfect nonlinear S-boxes. In: Davies, D.W. (ed.) *Advances in Cryptology — EUROCRYPT '91*. pp. 378–386. Springer Berlin Heidelberg, Berlin, Heidelberg (1991)
- [22] Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseth, T. (ed.) *EUROCRYPT'93*. LNCS, vol. 765, pp. 55–64. Springer, Heidelberg (May 1994)
- [23] Nyberg, K.: Linear approximation of block ciphers (rump session). In: Santis, A.D. (ed.) *EUROCRYPT'94*. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (May 1995)
- [24] Nyberg, K., Knudsen, L.R.: Provable security against differential cryptanalysis (rump session). In: Brickell, E.F. (ed.) *CRYPTO'92*. LNCS, vol. 740, pp. 566–574. Springer, Heidelberg (Aug 1993)
- [25] Sakiyama, K., Sasaki, Y., Li, Y.: *Security of Block Ciphers: From Algorithm Design to Hardware Implementation*. Wiley Publishing, 1st edn. (2015)
- [26] Shannon, C.E.: A mathematical theory of communication. *Bell System Technical Journal* 27(3), 379–423 (1948), <https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x>
- [27] Shannon, C.E.: Communication theory of secrecy systems. *The Bell System Technical Journal* 28(4), 656–715 (1949)
- [28] Stinson, D., Paterson, M.: *Introduction to Cryptography: Theory and Practice*, pp. 1–14 (08 2018)
- [29] Yu, Y., Wang, M., Li, Y.: A matrix approach for constructing quadratic APN functions. *Designs, Codes and Cryptography* 73 (11 2014)

Examples of 8-bit APN Function

Example 1

[0, 0, 0, 236, 0, 20, 164, 92, 0, 25, 100, 145, 179, 190, 115, 146, 0, 231, 122, 113, 105, 154, 183, 168, 119, 137, 105, 123, 173, 71, 23, 17, 0, 239, 131, 128, 29, 230, 58, 45, 213, 35, 50, 40, 123, 153, 56, 54, 148, 156, 109, 137, 224, 252, 189, 77, 54, 39, 171, 86, 241, 244, 200, 33, 0, 73, 32, 133, 72, 21, 204, 125, 197, 149, 129, 61, 62, 122, 222, 118, 14, 160, 84, 22, 47, 149, 209, 135, 188, 11, 130, 217, 46, 141, 180, 251, 62, 152, 157, 215, 107, 217, 108, 50, 46, 145, 233, 186, 200, 99, 171, 236, 164, 229, 125, 208, 152, 205, 229, 92, 195, 155, 126, 202, 76, 0, 85, 245, 0, 87, 77, 246, 49, 114, 216, 119, 139, 197, 162, 0, 9, 83, 132, 50, 11, 187, 60, 96, 83, 247, 192, 136, 247, 94, 164, 225, 28, 161, 235, 186, 195, 123, 13, 89, 239, 67, 133, 197, 157, 60, 55, 122, 2, 183, 12, 85, 92, 3, 232, 91, 25, 82, 9, 174, 117, 51, 165, 15, 131, 209, 247, 73, 123, 101, 22, 228, 2, 8, 203, 45, 53, 50, 60, 215, 255, 236, 82, 173, 126, 135, 105, 124, 110, 131, 221, 220, 71, 167, 52, 56, 228, 16, 51, 43, 134, 119, 104, 117, 226, 7, 168, 161, 29, 245, 151, 147, 202, 54, 228, 244, 23, 1, 131, 121, 26, 24, 42, 196, 251, 244, 11, 232, 69, 94, 17, 230]

Example 2

[0, 0, 0, 236, 0, 20, 164, 92, 0, 198, 100, 78, 108, 190, 172, 146, 0, 56, 165, 113, 182, 154, 183, 119, 168, 86, 105, 123, 114, 152, 23, 17, 0, 239, 92, 95, 194, 57, 58, 45, 213, 252, 237, 40, 123, 70, 231, 54, 148, 67, 109, 86, 224, 35, 189, 146, 233, 248, 116, 137, 241, 244, 200, 33, 0, 150, 32, 90, 72, 202, 204, 162, 26, 74, 94, 226, 62, 122, 222, 118, 14, 160, 139, 201, 240, 74, 209, 135, 188, 212, 93, 217, 46, 82, 107, 251, 62, 71, 66, 215, 180, 217, 108, 237, 241, 78, 233, 186, 23, 188, 171, 236, 164, 229, 125, 208, 152, 205, 229, 92, 195, 68, 126, 21, 147, 0, 138, 245, 0, 87, 146, 41, 49, 114, 7, 168, 139, 26, 125, 0, 214, 83, 132, 237, 212, 187, 227, 96, 83, 40, 192, 87, 247, 94, 164, 225, 28, 161, 235, 186, 28, 164, 210, 134, 239, 67, 133, 197, 66, 60, 232, 122, 221, 183, 211, 85, 92, 220, 55, 91, 25, 141, 214, 174, 170, 236, 165, 15, 131, 209, 40, 150, 164, 101, 22, 59, 221, 8, 203, 242, 53, 50, 227, 8, 32, 51, 82, 173, 126, 135, 105, 124, 177, 92, 2, 3, 71, 120, 52, 231, 228, 207, 51, 244, 134, 168, 104, 170, 61, 7, 119, 161, 194, 42, 72, 76, 21, 233, 59, 43, 200, 222, 131, 121, 197, 199, 42, 196, 36, 244, 11, 55, 69, 129, 206, 230]

Example 3

[0, 0, 0, 115, 0, 189, 79, 129, 0, 203, 125, 197, 106, 28, 88, 93, 0, 39, 37, 113, 149, 15, 255, 22, 141, 97, 213, 74, 114, 35, 101, 71, 0, 68, 133, 178, 110, 151, 164, 46, 126, 241, 134, 122, 122, 72, 205, 140, 225, 130, 65, 81, 26, 196, 245, 88, 18, 186, 207, 20, 131, 150, 17, 119, 0, 112, 145, 146, 8, 197, 214, 104, 234, 81, 6, 206, 136, 142, 43, 94, 204, 155, 120, 92, 81, 187, 170, 51, 171, 55, 98, 141, 92, 125, 218, 136, 34, 22, 54, 113, 68, 205, 31, 229, 182, 73, 223, 83, 186, 248, 156, 173, 15, 28, 62, 94, 252, 82, 130, 95, 22, 206, 90, 241, 143, 234, 140, 154, 0, 78, 165, 152, 121, 138, 147, 19, 227, 102, 59, 205, 240, 200, 103, 44, 33, 72, 161, 187, 205, 25, 2, 165, 79, 237, 178, 99, 201, 214, 123, 23, 112, 122, 80, 41, 103, 208, 8, 204, 237, 44, 176, 2, 144, 236, 130, 141, 176, 157, 181, 235, 50, 162, 120, 155, 160, 70, 216, 77, 72, 19, 127, 87, 92, 98, 104, 37, 45, 174, 86, 166, 85, 160, 28, 154, 78, 6, 72, 115, 177, 168, 160, 202, 85, 241, 11, 220, 53, 231, 89, 248, 187, 212, 152, 132, 14, 116, 191, 182, 17, 214, 239, 91, 121, 200, 181, 119, 12, 0, 143, 240, 2, 95, 150, 184, 136, 104, 83, 192, 248, 110, 17, 244, 24, 51, 190, 230]

Example 4

[0, 0, 0, 12, 0, 20, 48, 40, 0, 66, 171, 229, 156, 202, 7, 93, 0, 7, 122, 113, 99, 112, 41, 54, 157, 216, 76, 5, 98, 51, 131, 222, 0, 84, 99, 59, 247, 183, 164, 232, 110, 120, 166, 188, 5, 7, 253, 243, 158, 205, 135, 216, 10, 77, 35, 104, 109, 124, 223, 194, 101, 96, 231, 238, 0, 134, 145, 27, 103, 245, 198, 88, 234, 46, 208, 24, 17, 193, 27, 199, 85, 212, 190, 51, 81, 196, 138, 19, 34, 225, 98, 173, 186, 109, 202, 17, 212, 6, 38, 248, 68, 130, 134, 76, 80, 192, 9, 149, 92, 216, 53, 189, 31, 202, 151, 78, 236, 45, 84, 153, 6, 145, 37, 190, 105, 234, 122, 245, 0, 152, 28, 136, 175, 35, 131, 3, 21, 207, 162, 116, 38, 232, 161, 99, 1, 158, 103, 244, 205, 70, 155, 28, 137, 84, 68, 149, 217, 16, 36, 225, 201, 5, 182, 118, 145, 73, 222, 10, 178, 60, 102, 228, 118, 236, 146, 4, 86, 157, 83, 148, 109, 178, 88, 139, 176, 57, 30, 155, 23, 138, 137, 24, 229, 251, 104, 122, 45, 39, 144, 150, 26, 70, 60, 108, 78, 6, 88, 28, 177, 168, 70, 83, 26, 23, 221, 220, 211, 136, 143, 216, 228, 171, 136, 203, 248, 178, 22, 80, 199, 153, 25, 75, 105, 97, 44, 40, 202, 214, 191, 175, 50, 127, 166, 231, 110, 55, 202, 159, 62, 49, 1, 2, 254, 229, 241, 230]

Example 5

[0, 0, 0, 6, 0, 20, 164, 182, 0, 243, 142, 123, 179, 84, 153, 120, 0, 13, 122, 113, 105, 112, 183, 168, 157, 99, 105, 145, 71, 173, 23, 251, 0, 239, 105, 128, 247, 12, 58, 199, 213, 201, 50, 40, 145, 153, 210, 220, 148, 118, 135, 99, 10, 252, 189, 77, 220, 205, 65, 86, 241, 244, 200, 203, 0, 163, 32, 133, 72, 255, 204, 125, 197, 149, 107, 61, 62, 122, 52, 118, 228, 74, 190, 22, 197, 127, 59, 135, 188, 225, 104, 51, 46, 103, 94, 17, 212, 152, 157, 215, 107, 51, 134, 216, 196, 123, 3, 186, 200, 99, 171, 6, 164, 229, 151, 208, 114, 39, 229, 182, 41, 155, 148, 32, 76, 234, 85, 245, 0, 189, 167, 28, 49, 152, 50, 157, 139, 197, 162, 234, 9, 83, 132, 216, 11, 187, 214, 96, 83, 247, 42, 136, 29, 94, 78, 11, 246, 161, 1, 80, 195, 145, 13, 89, 5, 67, 111, 47, 157, 60, 221, 122, 232, 93, 12, 191, 92, 3, 232, 177, 243, 184, 227, 174, 159, 51, 165, 15, 131, 59, 29, 163, 123, 101, 252, 228, 2, 8, 33, 45, 53, 216, 60, 215, 255, 6, 82, 173, 148, 135, 105, 124, 132, 131, 221, 220, 71, 167, 52, 210, 228, 16, 51, 193, 108, 157, 130, 117, 226, 7, 168, 75, 247, 245, 151, 147, 202, 220, 14, 30, 23, 235, 131, 121, 240, 24, 192, 46, 17, 30, 11, 2, 69, 94, 251, 230]

Example 6

[0, 0, 0, 12, 0, 20, 48, 40, 0, 66, 171, 229, 156, 202, 7, 93, 0, 7, 122, 113, 99, 112, 41, 54, 4, 65, 213, 156, 251, 170, 26, 71, 0, 205, 250, 59, 110, 183, 164, 113, 247, 120, 166, 37, 5, 158, 100, 243, 158, 84, 30, 216, 147, 77, 35, 241, 109, 229, 70, 194, 252, 96, 231, 119, 0, 134, 145, 27, 254, 108, 95, 193, 234, 46, 208, 24, 136, 88, 130, 94, 204, 77, 39, 170, 81, 196, 138, 19, 34, 225, 98, 173, 35, 244, 83, 136, 212, 159, 191, 248, 68, 27, 31, 76, 201, 192, 9, 12, 197, 216, 53, 36, 134, 202, 151, 215, 117, 45, 84, 0, 159, 145, 37, 39, 240, 234, 122, 108, 0, 152, 133, 17, 175, 35, 26, 154, 21, 207, 59, 237, 38, 232, 56, 250, 1, 158, 254, 109, 205, 70, 2, 133, 16, 205, 68, 149, 64, 137, 36, 225, 80, 5, 47, 118, 145, 208, 222, 147, 178, 165, 102, 125, 239, 236, 11, 4, 207, 157, 202, 148, 109, 43, 88, 18, 41, 57, 135, 155, 23, 19, 137, 129, 124, 98, 104, 122, 45, 39, 9, 15, 131, 223, 60, 108, 78, 6, 193, 133, 177, 168, 223, 202, 131, 142, 221, 220, 74, 17, 143, 216, 228, 171, 17, 82, 248, 43, 22, 201, 199, 0, 25, 210, 240, 97, 181, 40, 83, 214, 38, 175, 171, 127, 63, 231, 247, 55, 83, 159, 167, 49, 152, 2, 103, 229, 104, 230]

Machine Configuration

```
> lscpu
Architecture:          x86_64
CPU op-mode(s):       32-bit, 64-bit
Byte Order:           Little Endian
CPU(s):               8
On-line CPU(s) list:  0-7
Thread(s) per core:   2
Core(s) per socket:   4
Socket(s):            1
NUMA node(s):        1
Vendor ID:            GenuineIntel
CPU family:           6
Model:               158
Model name:          Intel(R) Xeon(R) CPU E3-1245 v6 @ 3.70GHz
Stepping:            9
CPU MHz:             3713.071
CPU max MHz:         4100.0000
CPU min MHz:         800.0000
BogoMIPS:            7392.00
Virtualization:      VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            8192K
NUMA node0 CPU(s):   0-7

> cat /proc/meminfo
MemTotal:             32869744 kB
```