Jean-Francis MICHON
Pierre VALARCHER
Jean-Baptiste YUNÈS (Eds.)

Fonctions Booléennes:

BFCA

07

2007

Cryptographie, Applications

Boolean Functions:

Cryptography, Applications

3rd, INTERNATIONAL WORKSHOP, BFCA'07
PARIS, FRANCE, MAY 2007
PROCEEDINGS

PUBLICATION DES UNIVERSITÉS DE ROUEN ET DU HAVRE

BFCA'07

BFCA'07

Boolean Functions: Cryptography and Applications

Edited by

Jean-Francis Michon
Pierre Valarcher
Jean-Baptiste Yunès

Proceedings of the conference

organized at the
Université Paris Diderot, May 2–4th, 2007

by the
Laboratoire d'Informatique Fondamentale:
Fondements et Applications de Paris Diderot

Laboratoire d'Informatique, de Traitement de
l'Information et des Systèmes de Rouen

Laboratoire d'Algorithmique, Complexité
et Logique de Paris Est

Publication des Universités de Rouen et du Havre

Contents

PREFACE

Jean-Francis Michon[1], Pierre Valarcher[2] and
Jean-Baptiste Yunès[3]

The Meeting

The "Boolean Functions: Cryptography and Applications" international meeting took place on May 2-3th, 2007, in Paris, France. It was the third conference, in the field of Boolean functions, organized by the LITIS, University of Rouen, the LIAFA, University Paris Diderot and the LACL University of Paris-Est.

The main purpose of the conference was to create contacts between many different scientists working on Boolean functions, and that goal was reached. Approximatetly 30 participants came from many different countries over the world.
All papers were reviewed by two competent referees.
The organizers would like to give special thanks to Professor McGuire from Claude Shannon Institute, Ireland, and Professor Langevin from University of Toulon, France for their invited talk.

L'Atelier

L'atelier international "Fonctions Booléennes: Cryptographie et Applications" s'est tenu les 7 et 8 mars 2007, à l'Université

[1] Université de Rouen, LITIS, 76821 Mont Saint Aignan Cedex, France.
email: jean.francis.michon@univ-rouen.fr
[2] Université Paris-Est, LACL, 94500 Crteil, France.
email: valarcher@univ-paris12.fr
[3] LIAFA - Université Denis Diderot - Paris 7. 175 rue Chevaleret, F-75013 Paris, France. email: Jean-Baptiste.Yunes@liafa.jussieu.fr

J-F. Michon, P. Valarcher, J-B. Yunès (Eds.): BFCA'07

Paris Diderot, France. Il s'agissait de la troisième conférence sur le thème des fonctions Booléennes. BFCA'07 a été organisé conjointement par le LIAFA de l'Université Diderot de Paris, le LITIS de l'Université de Rouen et le LACL de l'Université Paris-Est.

Le but premier de cette conférence est de faire se rencontrer de nombreux chercheurs travaillant sur les fonctions Booléennes. Tous les articles ont été examinś par deux juges compétents. Nous tenons à remercier le professeur McGuire de l'Institut Claude Shannon, Irlande, et le professeur Langevin de l'Université de Toulon, France, pour leurs conférences invitées.

Thanks/Remerciements

Many thanks to our sponsors:
Un grand merci à nos sponsors:

Le LIAFA
L'Université Paris 7
Le LITIS
Le LACL
Le CNRS

Organizing committee/Comité d'organisation

Jean-Francis Michon (Univ. de Rouen, LITIS)
Pierre Valarcher (Univ. Paris-Est, LACL)
Jean-Baptiste Yunès (Univ. Paris 7, LIAFA)

Secretary/Secrétariat

Noëlle Delgado (LIAFA)
Louise Fauconnier (LIAFA)

Program committee/Comité de programme

Ali Akhavi (LIAFA/CNRS, France)
Hervé Chabanne (SAGEM, France)
Philippe Guillot (Univ. Paris 8, France)
Subhamoy Maitra (Indian Statistical Institute, Kolkata, India)
Jean-Francis Michon (LITIS, Rouen, France)
François Rodier (Institut de Mathématiques de Luminy, France)
Pierre Valarcher (LACL, Créteil, France
Melek D. Yücel (Middle East Technical University, Turkey)
Jean-Baptiste Yunès (LIAFA, Paris, France)

Referees/Examinateurs

Ali Akhavi
Lejla Batina
Hervé Chabanne
Mireille Fouquet
Éric Garrido
Philippe Guillot
Aline Gouget
Fabien Laguillaumie
Subhamoy Maitra
Jean-Francis Michon
Emmanuel Prouff
François Rodier
Damien Vergnaud
Melek Yücel
Jean-Baptiste Yunès

BFCA on the WEB/BFCA sur Internet

`http://www.liafa.jussieu.fr/bfca/`

Paris, September (Septembre), 2007

J-F. Michon, P. Valarcher, J-B. Yunès (Eds.): BFCA'07

Boolean Functions: Cryptography and Applications BFCA'07
Fonctions Booléennes : Cryptographie & Applications

IMPROVING SIDE-CHANNEL ATTACKS BY EXPLOITING SUBSTITUTION BOXES PROPERTIES

Sylvain Guilley[1,2], Philippe Hoogvorst[1,2], Renaud Pacalet[1,3] and Johannes Schmidt[4]

Abstract. This article revisits the "Correlation Power Attack" (CPA [18]), and justifies its physical relevance regarding CMOS circuits dissipation model. The CPA is then shown to be practical – and reproducible – on a real piece of hardware (DES co-processor.) Based on this successful attack, a theory about the vulnerability is derived. It happens that the attack asymptotic strength is not related to the acquisition conditions, but only to the algorithm implementation. In the case of an iterative implementation of a Feistel cipher, we show that the customarily used power models are valid. Within this theoretical framework, the attack strength depends only on the substitution boxes mathematical properties. A new distinguisher (9), more efficient than the transparency order [10], is proposed. Two enhancements of the proposed distinguisher are presented. The study of the relationship between the proposed distinguishers and the substitution boxes is still an open problem.

Key words: Security of hardware, side-channels analysis, attack algorithms, maximum likelihood evaluation, criteria on vectorial Boolean functions (substitution boxes, *aka* sboxes.)

[1] email: {guilley, hoogvorst, pacalet}@enst.fr
[2] GET/ENST, CNRS LTCI (UMR 5141), 46 rue Barrault, F-75634 Paris Cedex 13, France.
[3] GET/ENST, CNRS LTCI (UMR 5141), Institut Eurecom BP 193, 2229 route des Crêtes, F-06904 Sophia-Antipolis Cedex, France.
[4] email: johannes.schmidt@mpq.mpg.de. Max-Planck-Institute of Quantum Optics, Hans-Kopfermann-Straße 1, D-85748 Garching, Germany.

J-F. Michon, P. Valarcher, J-B. Yunès (Eds.): BFCA'07

1. Introduction

Electronic systems that embed cryptographic material are vulnerable to side-channel attacks. Every cryptographic implementation, be it software or hardware, leaks physical information about its internal state. More precisely, the usage of Boolean variables by complementary-MOS (CMOS [7]) circuits is responsible for charge transfers. The consequence is an observable power consumption and an electromagnetic field generation. Those dynamic quantities can be acquired by an attacker. They are rich in information because they are correlated with the manipulated data. Exploiting side-channels (power consumption, electromagnetic emissions, *etc.*) of hardware has proved to be a successful technique to acquire information about the key being used for ciphering.

Two categories of side-channel attacks can be defined, depending on their *modus operandi*.

(1) The so-called "template" attacks consist in a long off-line profiling step, that enables future fast on-line attacks.
(2) The so-called "correlation" attacks work as greedy algorithms: the side-channel information is analyzed until the secrets are extracted.

Template attacks [4, 9] require that a clone of the target attacked (or the target itself in open platforms) be available. This clone is then used as a training device, that is exercised in order to build up a side-channel database.

The on-line attack consists in matching the side-channel information acquired on the actual target device with that collected in the profiling preliminary stage. The correct key guesses are distinguished from the bad ones based on the analysis of the deviations from the profile database. The attack thus relies on a measurement-*versus*-measurement comparison.

The correlation attacks work differently: a known or suspected physical syndrome is looked for in the acquired side-channel information. The attack can thus begin from scratch. It ends as soon as the correlation with the physical syndrome overcomes a given signal-to-noise ratio, that makes it possible to decide unambiguously the correct values of the subkeys. Contrary to template attacks, correlation attacks require some *a priori* information about the architecture of the algorithm under attack: the attacker must indeed be able to devise a so-called "selection function", and to

access either the plaintext or the ciphertext. The goal of this function is to extract from the power traces only one relevant part. The extraction is consistent if the selection function is correlated to an actual internal dissipation occurring in the attacked chip; otherwise, it is decorrelated (at first order) and the extracted signal appears like noise. The term of "correlation attack" was first coined by É. Brier *et al.* in 2003 [17]. It made more clear the working factor of the original DPA from P. Kocher [6]: this seminal attack is indeed a single-bit correlation attack in the particular case when the sensitive data is used right after a constant (plaintext-independent) operation.

All these attacks have been shown to be practical on unprotected implementations. It is now to be feared that they improve in such a way they become able to defeat protected implementations as well. Unfortunately, this scenario is all the more likely as neither the template nor the correlation attacks are optimal. As a matter of fact, the template attacks do not exploit the knowledge of the underlying implementation, and correlation attacks do not use a clone device to devise a fine-tuned power dissipation model.

The strategy presented in this paper consists in using the advantages of both the template and the correlation attacks. The goal of this paper is to show how the use of a model of the exploited dissipation, possibly extracted from a clone device, can enhance the attack. In particular, the goal is not to demonstrate the fastest possible attack. For this reason, plain traces, without any signal processing, are used. In addition, we do not take advantage of any peculiarity of the design under analysis: so, to remain consistent, we present a basic register transfer attack (although tailored attacks might be more powerful.)

The rest of the article is organized as follows. Section 2 presents the correlation power attack based on a CMOS power model. The goal of this section is to provide a didactic explanation of this attack, illustrated on the example of a DES [8] co-processor. Section 3 provides experimental evidences that the CPA works when applied on real-life encryptions. The choice for a selection function based on a Hamming distance (HD) is motivated here. In the section 4, a theory for the CPA is presented. This theory merges results from the original CPA [18] and from the key hypotheses disambiguation using an maximum likelihood estimator (MLE [11].) A new criterion, namely Equation (9), for the power

attack strength is proposed in Sec. 4.3.3. In Sec. 5, two optimizations for the attack are presented. It is an open issue to find links and to compare these criteria. Finally, Sec. 6 concludes the paper and emphasizes that challenging open problems related to Boolean functions are presented in this paper. The appendices A and B provide detailed technical information about the attacked circuit and the acquisition setup. The appendix C provides with the detailed proofs of some lemmas.

2. Correlation Power Attack

2.1. Power Model of CMOS Circuits

In the historical DPA of P. Kocher [6], no explicit link was explained between the power curves and the gates dissipation: the attack only assumed a "mysterious" bias. This section explains the nature of the leaks in two popular power models: Hamming weight and Hamming distance [17].

In CMOS circuits [7], logic gates only leak information when their output toggles. This information can be collected by an attacker as a current intensity (power attack), a radiated electromagnetic field (EM attack), or any other auxiliary physical channels. In the sequel, we focus on power attacks, where an attacker monitors the device's activity thanks to acquisition of the voltage drop across a "spying" resistance malevolently placed between the power supply source and the power input of the targeted device. Depending on the relative N (negatively doped) and P (positively doped) MOS transistors dimensions and on the capacitive environment of the net it loads, the energy can be different whether the output rises or falls. We denote these quantities with $\xi^{\uparrow}$ and $\xi^{\downarrow}$. The overall chip consumption is thus made up of the accumulation of the individual contributions from all the gates. In a cryptoprocessor, the inputs are the message m and the key k. In addition, if the implementation is synchronous, the gates only change states consecutively to a rising edge of the global clock. The chip power consumption occurring at period $t \rightarrow t+1$ is thus equal to:

$$\text{power} \doteq \sum_{i \in \text{nets}} \xi_i^{\uparrow} \underbrace{\overline{i}(t) \cdot i(t+1)}_{\text{Net } i \text{ has a rising edge}} + \xi_i^{\downarrow} \underbrace{i(t) \cdot \overline{i}(t+1)}_{\text{Net } i \text{ has a falling edge}} . \quad (1)$$

We do not claim that this model is original: it is for instance used in [11], and also in the basic power analysis engines embedded in CAD tools, such as Cadence [2].

Concretely, the consumption defined in (1) is perturbed by some sources of noise:

- first of all, the gates $i \neq i'$ consumption is not totally decorrelated, due to cross-talk between nets,
- second, the combinatorial parts are incurred by glitches,
- third, the chip environment might vary during the acquisition, and
- fourth, the acquisition apparatus brings its own imprecision, for instance due to quantification noise.

2.2. **Side-Channel Information Extraction**

The power model (1) provides an integrated information about the circuit's activity. In the context of side-channel analysis, the attacker wishes to extract the activity of a single net. We place ourselves in the case where the attacker knows the exact functionality of the circuit, but not its layout. She is thus able to acquire traces, and to weight them with a "selection function", noted S. This function (for a single target net j) can be defined as:

(1) the Hamming weight (HW): $j(t+1)$ or
(2) the Hamming distance (HD): $j(t) \oplus j(t+1)$.

It is preferable to use the ± 1 "signed" versions of those functions (thus balanced), because the residual noise is averaged to zero. The selection functions S are thus:

(1) the balanced Hamming weight: $(-1)^{j(t+1)}$ or
(2) the balanced Hamming distance: $(-1)^{j(t)\oplus j(t+1)}$.

In a typical cryptographic algorithm (such as in a product block cipher), the successive intermediate data are crafted to be as decorrelated as possible from each other. If "$\mathbb{E}$" denotes the expectation of a random variable, it is thus reasonable to assume that:

$$\begin{aligned} \mathbb{E}(i(t) \cdot i(t+1)) = \mathbb{E}(i(t) \cdot \overline{i(t+1)}) &= \\ \mathbb{E}(\overline{i(t)} \cdot i(t+1)) = \mathbb{E}(\overline{i(t)} \cdot \overline{i(t+1)}) &= \left(\frac{1}{2}\right)^2 = \frac{1}{4}\,. \end{aligned}$$

Now, using the identities $(-1)^x = 1 - 2 \cdot x$, $\overline{x} = 1 - x$ and $\mathbb{E}(x) = \frac{1}{2}$, for all $x \in \{0, 1\}$, it is easy to compute the average

signal got by an attacker using the two latter selection functions (with "power" being equal to the random variable defined in (1)):

$$\begin{cases} \mathbb{E}\left(\text{power} \times \left(-2 \cdot (-1)^{j(t+1)}\right)\right) &= \frac{\xi_j^{\uparrow} - \xi_j^{\downarrow}}{2}\,, \\ \mathbb{E}\left(\text{power} \times \left(-2 \cdot (-1)^{j(t)\oplus j(t+1)}\right)\right) &= \frac{\xi_j^{\uparrow} + \xi_j^{\downarrow}}{2}\,. \end{cases} \tag{2}$$

The detailed demonstration is given in appendix C.1 at page 23. As the target gate dissipates power on both types of transitions, $\xi_j^{\uparrow}$ and $\xi_j^{\downarrow}$ are strictly positive for all the nets j in the netlist.

Moreover, it is worth restricting our study to attacks on the sequential elements (DFFs in synchronous circuits.) In this case, banks of registers are activated simultaneously (by a global clock), which allows for the coherent summation of their individual contribution. The registers contain data that depend on some bits of the key. Given one hypothesis, the prospective value of many internal nodes can be guessed. It is thus relevant to attack those bits together, using multi-bit selection functions. In addition, we assume that at the end of a round the bits of a word are made as much independent as possible. This independence hypothesis is not exactly true, but it allows for a simple model. Under the assumption that: $\forall i \neq j, \quad \mathbb{E}(i \cdot j) = \mathbb{E}(i) \cdot \mathbb{E}(j) = \frac{1}{4}$, the multi-bit correlation yields:

$$\begin{cases} \mathbb{E}\left(\text{power} \times \left(-2 \cdot \sum_{j\in J} (-1)^{j(t+1)}\right)\right) &= \frac{\sum_{j\in J} \xi_j^{\uparrow} - \xi_j^{\downarrow}}{2}\,, \\ \mathbb{E}\left(\text{power} \times \left(-2 \cdot \sum_{j\in J} (-1)^{j(t)\oplus j(t+1)}\right)\right) &= \frac{\sum_{j\in J} \xi_j^{\uparrow} + \xi_j^{\downarrow}}{2}\,. \end{cases} \tag{3}$$

To the authors' knowledge, these two equations provide the first formal justification of the CPA.

In [15], Thomas Messerges discusses an attack on a software implementation of DES based on the guess of the substitution boxes output. In this work, the observed peaks are explained by the number of transitions in a register. However, if the assembly of the code being executed is not known to the attack (which was the case for Messerges), the exact sequence of instructions executed is also unknown. For this reason, the power model is taken equal to the Hamming weight of the substitution boxes (*aka* sboxes.) The motivation for this choice is that, by chance, the content of the register at the previous clock can happen to be constant

(independent of the data.) In this case, the Hamming distance model simplifies into a Hamming weight model.

Notice that apparently different selection functions can yield correlated peaks. A relevant example is to consider: $\overline{j(t)} \oplus j(t+1)$, instead of $j(t) \oplus j(t+1)$. The resulting differential is exactly the opposite, because for an n-bit word w, $|w| - n/2 = n/2 - |\overline{w}|$. Now, with $w = j(t) \oplus j(t+1)$, $\overline{j(t)} \oplus j(t+1) = \overline{j(t) \oplus j(t+1)} = \overline{w}$. A good evaluator for the correlation between two selection functions x and y is the so-called Pearson correlation:

$$\frac{\mathbb{E}\left((x - \mathbb{E}x) \cdot (y - \mathbb{E}y)\right)}{\sqrt{\mathbb{E}(x - \mathbb{E}x)^2} \cdot \sqrt{\mathbb{E}(y - \mathbb{E}y)^2}}\,.$$

In the case of $x = |w|$ and $y = |\overline{w}|$, the Pearson correlation is maximal in absolute value (it is equal to $|-1|$.)

2.3. **Information Extraction Limitations**

If, instead of (1), a static-leakage aware power dissipation model (denoted power$'$) is used:

$$\begin{aligned} \text{power}' \doteq \sum_{i \in \text{nets}} \quad & \xi_i^{00} \overline{i(t)} \cdot \overline{i(t+1)} + \xi_i^{01} \overline{i(t)} \cdot i(t+1) + \\ & \xi_i^{10} i(t) \cdot \overline{i(t+1)} + \xi_i^{11} i(t) \cdot i(t+1)\,, \end{aligned} \tag{4}$$

we show that it is possible for no attacker to extract neither the pure static leakage (such as ξ_j^{00}) nor the pure dynamic leakage (such as $\xi_j^{01} = \xi_j^{\uparrow}$.) As a matter of fact, any selection function S that involves nets states at times t and/or $t+1$ can be expressed as:

$$\begin{aligned} S \;\doteq\; & y_j^{00} \overline{j(t)} \cdot \overline{j(t+1)} + y_j^{01} \overline{j(t)} \cdot j(t+1) + \\ & y_j^{10} j(t) \cdot \overline{j(t+1)} + y_j^{11} j(t) \cdot j(t+1)\,, \end{aligned}$$

where $\left(y_j^{00}, y_j^{01}, y_j^{10}, y_j^{11}\right) \in \mathbb{R}^4$ are four numerical constants chosen by the attacker. The mathematical expectation of the product power$' \times S$ is proportional to: $\xi_j^{00} y_j^{00} + \xi_j^{01} y_j^{01} + \xi_j^{10} y_j^{10} + \xi_j^{11} y_j^{11}$. The extraction of ξ_j^{00} or ξ_j^{01} is impossible if $y_j^{00} + y_j^{01} + y_j^{10} + y_j^{11} = 0$. This condition is however necessary for the interference with extraneous nets $i \neq j$ to be cancelled.

2.4. Hypothesis Testing and Key Cracking

It is now possible to sketch the scenario for a power attack. When the attacked circuit performs an encryption, the plaintext might be known, but not the intermediate results after the first round. The nets whose activity is relevant to be extracted are those from the datapath. If an encryption begins at time t, then the plaintext $j(t)$ is assumed to be known. The value of the datapath register at time $t+1$ depends on $j(t)$ and on the first round key. However, in most block ciphers, be them Feistel or substitution-permutation networks, the round keys are injected into the data as small chunks. For example, $j(t+1)$ depends:

- on 4 bits of the key for Serpent,
- on 6 bits of the key for DES,
- on 8 bits of the key for AES, SKIPJACK, KHAZAD and SAFER.

An attack thus consists in testing all the possible selection functions: for every chunk of the key, there are 2^4, 2^6 or 2^8 of them for the abovementioned popular algorithms. The correct selection function will exhibit the bias computed in (3). The incorrect selection functions are expected to be decorrelated from the power traces, and thus to exhibit *no* or *little* bias.

3. Experimental Validation of CPA Attacks

3.1. Attacks Reproducibility

Prior to developing a theory about CPA, we need to be confident in the fact that attacks are indeed reproducible. For this purpose, two experimental conditions are evaluated on a DES cryptoprocessor:

- **(Setup 1)**at nominal voltage 1.2 V, with a spying resistor R of 11 Ω [13],
- **(Setup 2)**the circuit is under-powered (V=0.7 volts) and a resistor of higher value (R=80 Ω) is used.

The circuit's dissipation profile is highly dependent on the experimental conditions, as shown in Fig. 1. In both cases, the plaintext x =`000011fdca19fd46` is encrypted with the key k =`6a65786a-65786a65`, resulting in the ciphertext y =`78ec7f6ff219a7fe`. The figure represents accurate measurements, at 20 Gsample/s, of the

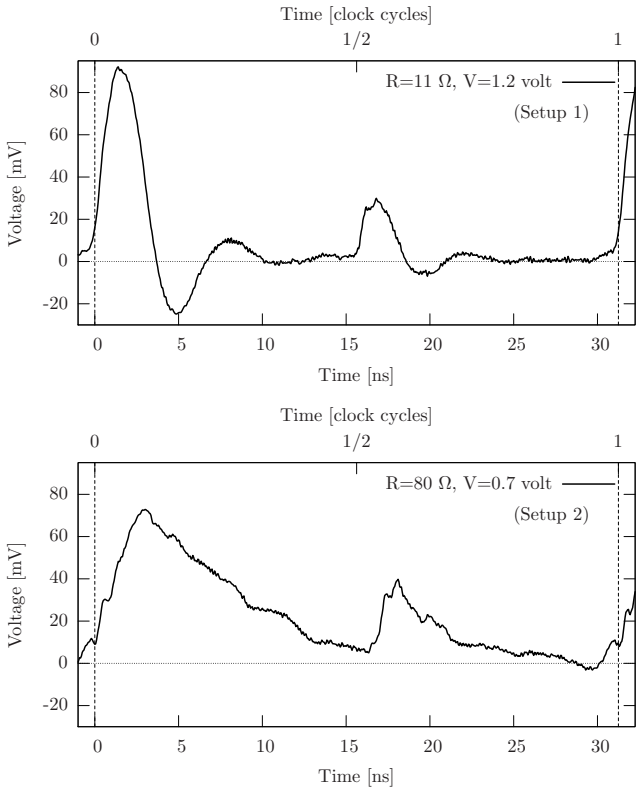


FIGURE 1. Trace power signature for two different environmental conditions.

(same) first round of DES, running at 32 MHz (hence a period of 31.25 ns.) The power signature is described below:

- the rising edge of the clock is responsible for the dissipation between 0 ns and 31.25 ns / 2, whereas
- its falling edge is responsible for the dissipation in the second half of the period.

The CPA is realized on those two types of traces. The working factor that is selected to quantify the attack success is a mere signal-to-noise ratio (SNR):

- the *signal* is the extraction of the datapath for the correct key guess, using extraction method presented in (3) with set J being the 64-bit LR register of DES, whereas

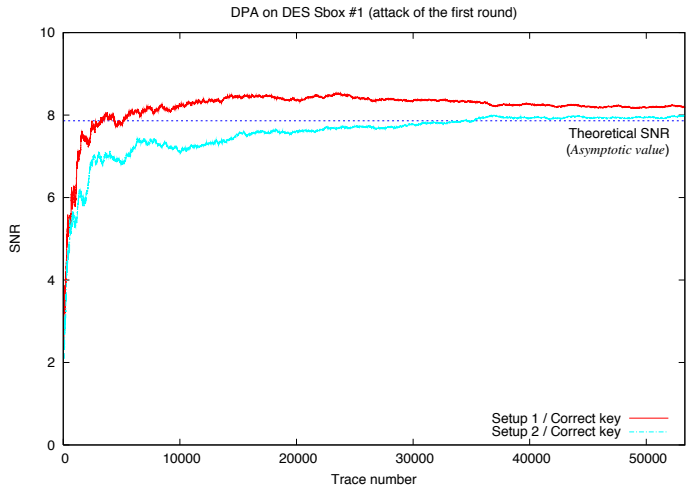


FIGURE 2. Evolution of the SNR of the DPA on DES sbox #1 with the number of accumulated traces.

- the *noise* is the standard deviation of the extractions for the incorrect key guesses.

Definition 3.1. Signal **Sig** *SNR*.

$$SNR(\mathbf{Sig}) \doteq \frac{\mathbf{Sig}(k=\dot{k})-\overline{\mathbf{Sig}}}{\left(\frac{1}{\#k-1}\sum_{k\neq\dot{k}}\left(\mathbf{Sig}(k)-\overline{\mathbf{Sig}}\right)^2\right)^{1/2}},$$

where $\overline{\mathbf{Sig}}$ is the signal mean, estimated as $\frac{1}{\#k}\sum_k \mathbf{Sig}(k)$.

The evolution of the SNR with the number of power traces (similar to the representative ones shown in Fig. 1) is given in Fig. 2 for the first sbox of DES.

Notice that a more elaborate criterion will be presented in Sec. 4. It will be based on a model, that makes it possible to derive a theoretical value for the SNR.

The model and the experimental SNRs are shown in Fig. 3 and appear to match when the actual key used during encryption is equal to the correct key $\dot{k}$. The asymptotic values are however not strictly identical, although a dependency in the sbox is clear. One possible explanation for this second-order discrepancy is suggested in Sec. 5.

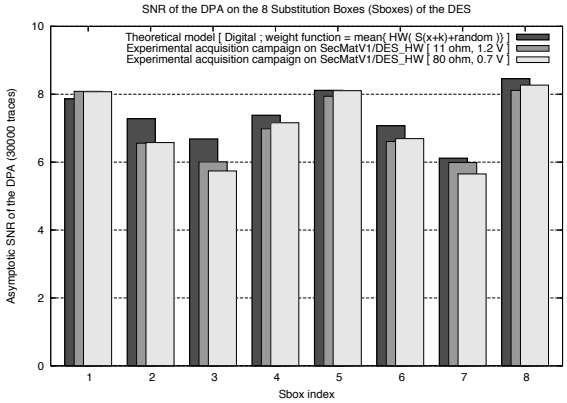


FIGURE 3. Comparison between theoretical *digital* model presented in Sec. 4 and experimental *analog* measurements of the DPA Signal-to-Noise Ratio (SNR) on the secret key encryption algorithm DES [8] embarked in the SecMat V1 ASIC (*cf.* the layout of *Fig. 7(a)*).

3.2. **Hamming Weight versus Hamming Distance**

As for DES, there are two common ways to attack:

(1) either known plaintext attacks, where the observer considers the first round of the encryption,
(2) or known ciphertext attacks on the last round of the DES algorithm.

We will elaborate on the former, but all assumptions and equations hold for the latter in analogy.

Figure 4 shows differential traces obtained by the weighting of traces with the selection function $\mathrm{HD}_J(t) \doteq \sum_{j\in J}(-1)^{j(t)\oplus j(t+1)}$ where J is the right 32-bit word (R) of DES datapath and $t = 0$ is the beginning on the encryption. Using NIST notations [8], the selection function $\mathrm{HD}_{\mathrm{R}}(0)$ is also expressed as $32 - 2 \times |\mathrm{R}_0 \oplus \mathrm{R}_1|$. This selection function extracts the number of transitions between the right half of the IP-permuted plaintext and the right half of the output of the first round. In figure 4, the extraction is actually plotted with two curves: the rising (*resp.* falling) edge selection function is $16 - 2 \times |\overline{\mathrm{R}_0} \cdot \mathrm{R}_1|$ (*resp.* $16 - 2 \times |\mathrm{R}_0 \cdot \overline{\mathrm{R}_1}|$.) Notice that the arithmetic sum of these curves is equal to $\mathrm{HD}_{\mathrm{R}}(0)$, because $|\mathrm{R}_0 \oplus \mathrm{R}_1| = |\overline{\mathrm{R}_0} \cdot \mathrm{R}_1| + |\mathrm{R}_0 \cdot \overline{\mathrm{R}_1}|$.

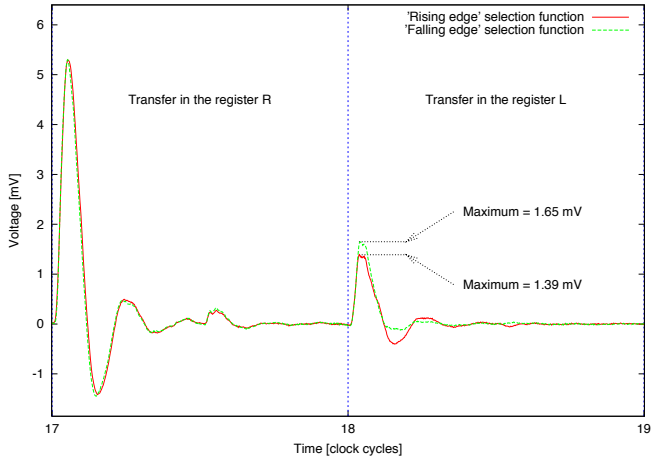


FIGURE 4. Differential traces resulting from the weighting by the two selection functions $16 - 2 \times \left|\overline{\mathrm{R}_0} \cdot \mathrm{R}_1\right|$ (*rising edge*) and $16 - 2 \times \left|\mathrm{R}_0 \cdot \overline{\mathrm{R}_1}\right|$ (*falling edge*) on DES.

At clock period 1, the transition occurs in the register R of DES, while at clock period 2, the transition occurs in the register L, because $|\mathrm{R}_0 \oplus \mathrm{R}_1| = |\mathrm{L}_1 \oplus \mathrm{L}_2|$. The power curves at clock period 1 show that: $\xi_{\mathrm{R}}^{\uparrow} \approx \xi_{\mathrm{R}}^{\downarrow} \approx 5.2$ mV, where $\xi_{\mathrm{R}}^{\uparrow,\downarrow} \doteq \sum_{j \in \mathrm{R}} \xi_{\mathrm{j}}^{\uparrow,\downarrow}$. The register R seems to be well balanced. As for the register L, the dissymmetry is significative: $1.39 \text{ mV} = \xi_{\mathrm{L}}^{\uparrow} < \xi_{\mathrm{L}}^{\downarrow} = 1.65$ mV. The origin of the discrepancy between the signature of registers R and L is not understood yet.

The Hamming distance model is thus much better, since it exploits a larger bias.

4. CPA Attacks Modelization

4.1. From Practice to Theory

The SNR of the CPA on DES revealed that the eight sboxes were not of equal strength. In this section, we seek an explanation for this observation. First of all, we must get rid of the dependency in the number of traces.

In practice, the traces are not random variables, such as in (1), but rather functions $T(x)$ of the ciphertext x. When few

traces (say N) are processed, the selection function S is biased. In experiments, the following computation is done, so as to make up for the bias: $\sum_{x=0}^{N-1} CPA(x) \neq \sum_{x=0}^{+\infty} CPA(x) = 0$ [12]. The correlation is computed as follows:

$$\mathrm{tr}(S \times T) - \mathrm{tr}(S) \times \mathrm{tr}(T)\,, \tag{5}$$

where "tr" is the *trace* operator: $\mathrm{tr}f \doteq \sum_x f(x)$. To simplify the model, we assume that enough samples are collected for the plaintexts to be equiprobable.

For the sake of clarity, the rest of the explanations are done using the unweighted Hamming Distance (HD) model. This means that when attacking a multi-bit register J, all $\xi_j^{\{\uparrow,\downarrow\}}$, for $j \in J$, are assumed to be equal. This quantity is thus a mere measure of transitions count. Only in the last Section 5 they will be reintroduced to demonstrate an improvement of the attack.

4.2. From Crypto-Systems to Sboxes

The existing correlation models are often discussed in terms of sboxes [10,12]. When attacking an entire crypto-system, the model must be adapted. Figure 5 shows the datapath involving the first sbox in an iterative hardwired DES implementation. The known plaintext is $j(t) \in J$, where J is the 64-bit register LR. Given the iterative nature of the algorithm, after the first round, $j(t+1)$ is overwriting $j(t)$ in the same register. However, it happens that the sub-set of J involved by the first sbox is:

- $j(t) \quad \in \mathrm{R}\{32,1,2,3,4,5\}$ in the IP'ed plaintext, and
- $j(t+1) \in \mathrm{R}\{9,17,23,31\}$ after the first round.

As $\{32,1,2,3,4,5\} \cap \{9,17,23,31\} = \emptyset$, $j(t)$ is independent from $j(t+1)$ when analyzing the first sbox. The same remark actually holds for all the sboxes. This property results from the *diffusion* of DES. Any Feistel cipher is expected to feature the same property. In this case, the attacker can decide to choose $j(t) = 0$, in which case $j(t) \oplus j(t+1) = j(t+1)$. Consequently, both the Hamming weight and the Hamming distance selection functions can be studied in a common framework.

4.3. Multi-bit Correlation Power Attack (CPA)

Emmanuel Prouff defined in [10] the *transparency order*, a metric of the vulnerability of sboxes against a certain class of power

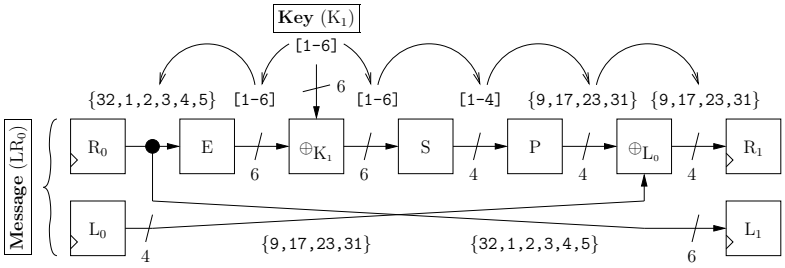


FIGURE 5. Datapath of DES involved in the DPA attack of the first round on sbox #1.

attacks. The considered attack scenario is a Hamming weight prediction of the sboxes outputs.

This section elaborates on this result, by considering the same selection function (multi-bit correlation power-analysis – CPA), but using a maximum likelihood estimator (MLE) to distinguish the correct key from the wrong hypotheses [11]. The new criterion (9), that has not been studied yet, is proposed as a metric to quantify the intrinsic strength of the targeted sbox.

4.3.1. Differential Traces

The target cryptographic function is: $x \mapsto f(x \oplus \dot{k})$, the output of a substitution box f, where the key $\dot{k}$ is injected via one `XOR`[1]. The vectorial Boolean function f operates from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. The Boolean coordinate $b \in [0, m[$ of f is denoted f_b. The attack model is the following: the power traces are expected to contain the scalar information $p(x) \doteq \sum_{b=0}^{m-1} f_b(x \oplus \dot{k})$, where x varies from trace to trace and where $\dot{k}$ is an unknown constant (*e.g.* a key.)

Note:

- *The power model can be extended to a parametrized leakage* $\langle \alpha | f \circ \tau_{\dot{k}} \rangle$*, where* $\alpha \in \mathbb{R}^m$ *models loads for each coordinate, and where* τ_k *is the* translation *of vector* k*:* $\tau_k(\cdot) \mapsto k \oplus \cdot$*. The studied case corresponds to* $\alpha = (1, \cdots, 1)$*. Refer to Sec. 5.2.*
- *Still better, a physical model that encompasses "signal integrity" issues, such as "cross-talk" between neighbor nets, can be used:* $\langle f \circ \tau_{\dot{k}} | \alpha | f \circ \tau_{\dot{k}} \rangle$*, where* $\alpha \in \mathbb{R}^{m \times m}$ *models the cross-talk (symmetrical) matrix. Diagonal terms* $\alpha_{b,b}$

[1]Other types of injection would yield the same results.

of matrix α are the components of vector α in the previous model without cross-talk, because $\alpha_{b,b}\big(f_b \circ \tau_{\dot{k}}\big)^2 = \alpha_b f_b \circ \tau_{\dot{k}}$. The studied case corresponds to $\alpha = Id_m$.

However, these considerations are only useful to fine-tune an attack to a given hardware. In the rest of this section, we suppose that the leak is perfect: all the bits sign with the same amount and they are not physically correlated.

The attacker correlates the traces with the collection of multi-bit selection functions $s_k : x \mapsto \sum_{b'=0}^{m-1} (-1)^{f_{b'}(x \oplus k)}$, indexed by $k \in \mathbb{F}_2^n$. For the sake of commodity, the key guess k is better off be taken relative to the actual key $\dot{k}$; the distance $\varepsilon \doteq k \oplus \dot{k}$ is thus considered in the sequel. A selection function must be balanced, for decorrelated contributions to average to zero. After having weighted enough traces, the attacker finally has at her disposal $\#\mathbb{F}_2^n = 2^n$ figures, namely:

$$\begin{array}{rl} & \sum_x \left(\sum_b f_b(x \oplus \dot{k})\right) \times \left(\sum_{b'} (-1)^{f_{b'}(x \oplus k)}\right) \\ = & \mathrm{tr} \sum_{b,b'} f_b (-1)^{f_{b'} \circ \tau_\varepsilon} = -\frac{1}{2} \mathrm{tr} \sum_{b,b'} (-1)^{f_b \oplus f_{b'} \circ \tau_\varepsilon} . \end{array} \tag{6}$$

The last equality is a consequence of the selection function being balanced. Indeed, if p is the power model ($\mathrm{tr}(p) > 0$, otherwise the cryptographic engine violates the second law of thermodynamics) and s the selection function, then $\mathrm{tr}(s) = 0 \Leftrightarrow \mathrm{tr}(ps) = \mathrm{tr}((p - \mathrm{tr}(p))s)$. This means that the same information can be extracted from plain p or from its centered variant $p - \mathrm{tr}(p)$.

Notice that if the target function was not $x \mapsto f(x \oplus \dot{k})$ (*cf* Sec. 4.3.1) but:

$$x \mapsto y \oplus f(x \oplus \dot{k}) = j(t+1), \tag{7}$$

where the initial state is $y \doteq j(t)$, then (6) would still be valid. The reason is that the contribution of y (for DES sbox #1, $y = \mathrm{R}_0\{9, 17, 23, 31\}$) is cancelled by the `XOR`ing between $j(t)$ and $j(t+1)$.

In addition, the quantity (6) is negative because, in case of an happy guess (*i.e.* $\varepsilon = 0$), the figure of merit for the matches $f_b(x \oplus \dot{k})$ *versus* $(-1)^{f_b(x \oplus \dot{k})}$ ($\forall b \in [0, m[$) is either $0 \cdot (-1)^0 = 0 \leq 0$ or $1 \cdot (-1)^1 = -1 \leq 0$. Correct hypotheses are thus statistically acknowledged by a negative weight.

For this reason, we define the differential traces as (twice) the opposite of the expression (6):

$$\Delta(\varepsilon) \doteq \mathrm{tr} \sum_{b,b'} (-1)^{f_b \oplus f_{b'} \circ \tau_\varepsilon} \,. \tag{8}$$

4.3.2. Ghost Peaks

The differential traces (8) have two remarkable properties:

(1) $\mathrm{tr}\Delta = \sum_\varepsilon \Delta(\varepsilon) = 0$ if f is balanced (this is our assumption in the sequel),

(2) $\Delta(0) = \mathrm{tr}\left(\sum_b (-1)^{f_b} \times \sum_{b'} (-1)^{f_{b'}}\right) = \mathrm{tr}\left(\sum_b (-1)^{f_b}\right)^2$. Consequently, $\Delta(0) \geq \Delta(\varepsilon)$, because the differential traces are the auto-correlation of the centered Hamming weight of the function f. The extensive proof of this property is given in appendix C.2 at page 23. This result is the first formal demonstration that the CPA is indeed a distinguisher between hypotheses on keys.

As, in our case, the correct selection function is equal to $p - \mathrm{tr}(p)$, $\Delta(0) = \mathrm{tr}(p^2) > 0$ (because $\mathrm{tr}(p) > 0$.) As a consequence, $\exists \varepsilon \neq 0$ such that $\Delta(\varepsilon) \neq 0$. The set $\{\Delta(\varepsilon), \varepsilon \neq 0\}$ is referred to as *ghost peaks*.

Given the second property, the correct key can be guessed by choosing the largest differential trace. This way of validating the hypothesis $k \stackrel{?}{=} \dot{k}$ leads to the *transparency order* $\mathcal{T}_f$ [10].

As already mentioned in equation (7), in an iterative hardwired implementation of DES, the initial state y to be replaced *in situ* by the sbox output depends neither of the plaintext x nor on the secret key $\dot{k}$. In this case, an attacker can choose the "*precharge state*" y to be equal to 0. Under this assumption, the transparency order can be expressed as:

$$\begin{aligned} \mathcal{T}_f &= \frac{1}{2^n - 1} \sum_{\varepsilon \neq 0} \left(|\Delta(0)| - |\Delta(\varepsilon)|\right) \\ &= |\Delta(0)| - \frac{1}{2^n - 1} \sum_{\varepsilon \neq 0} |\Delta(\varepsilon)| \,. \end{aligned}$$

4.3.3. MLE as Hypotheses Test

The previous key candidates disambiguation is sub-optimal, because it treats ghost peaks as noise, although they are predictable.

The MLE method described in R. Bévan's PhD thesis [11] consists in computing a distance between full constellation of ghost peaks and the expected constellation. Notice that in R. Bévan's work, the correlations are not computed explicitly: this section develops R. Bévan computations.

We consider in the sequel the Euclidean distance $||\cdot||_2$, but other metrics could be more suitable (especially if the power model is weighted by different real coefficients.)

The attacker thus computes the following set of distances, indexed by ε:

$$||\overrightarrow{\Delta \circ \tau_\varepsilon} - \overrightarrow{\Delta}||_2 \,,$$

where $\overrightarrow{f} = (f(0), f(1), \cdots, f(2^n - 1))$ is the vector made up of function f values (*i.e.* its truth table represented flattened.)

This quantity can be expanded as:

$$\begin{aligned} & \textstyle\sum_e \left(\mathrm{tr} \sum_{b,b'} (-1)^{f_b} \cdot \left((-1)^{f_{b'} \circ \tau_{e \oplus \varepsilon}} - (-1)^{f_{b'} \circ \tau_e} \right) \right)^2 \\ = & \textstyle\sum_e \left(\mathrm{tr} \sum_{b,b'} (-1)^{f_b \circ \tau_e} \cdot \left((-1)^{f_{b'} \circ \tau_\varepsilon} - (-1)^{f_{b'}} \right) \right)^2 . \end{aligned}$$

Thus, the attack will be all the more easy as the following metric is high:

$$\boxed{\min_{\varepsilon \neq 0} \sum_e \left(\mathrm{tr} \sum_{b,b'} (-1)^{f_b \circ \tau_e} \cdot \left((-1)^{f_{b'} \circ \tau_\varepsilon} - (-1)^{f_{b'}} \right) \right)^2} . \qquad (9)$$

5. Attacks Enhancement Proposals

5.1. Multi-Dimensional Selection Function

The attacker can also guess the output bits b' one by one, yielding in bitwise differential traces:

$$\forall b' \in [0, m[, \quad \Delta(b', \varepsilon) \doteq \mathrm{tr} \sum_b (-1)^{f_b \oplus f_{b'} \circ \tau_\varepsilon} . \qquad (10)$$

After that, the attacker simply computes a distance in an $m \times 2^n$-dimensional space.

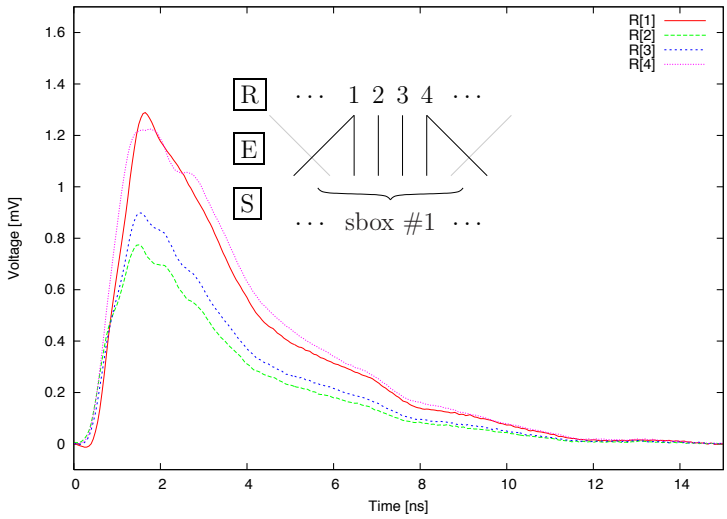


FIGURE 6. Extraction of the power consumption of the four output bits of DES sbox #1.

5.2. **Weighted Power Model**

All the bits of the target register are not identical, from a physical point of view. Figure 6 shows the values $\frac{1}{2}\left(\xi_j^{\uparrow} + \xi_j^{\downarrow}\right)$ extracted for each of the four bits $j \in \{1, 2, 3, 4\}$ of the first sbox output.

It clearly appears that bits 1 & 4 sign with a greater intensity than bits 2 & 3. Knowing the architecture of DES, the reason is straightforward: the expansion E of DES induces one extra fanout for the extremal bits R[1] and R[4]. An attacker can take advantage of this *a priori* information (either extracted from the layout or characterized on the device itself) to fine-tune her attack.

6. **Conclusion**

Correlation power attacks have been applied with success on real devices. This tool allows to extract *local* information out of *global* execution traces. The attack consists in testing an hypothesis on a sub-key, involved in the extraction. Some seminal works, by E. Prouff [10] and C. Carlet [3], model the CPA attack, and prove that its strength is directly correlated to cryptanalytic

properties of the substitution boxes featured by symmetrical block cipher algorithms. However, the attack strategy is not optimal, in the sense that the best hypothesis is selected, thus disregarding the structure of the false hypotheses. R. Bévan's proposed in [11] an optimal key hypothesis test, based on a maximum likelihood estimator. This strategy is explicited in this paper. It leads to the proposal for a new criterion (9) to quantify the weakness of an sbox in front a CPA. This criterion opens up a new field of investigations, such as trade-offs between mandatory cryptographic properties of sboxes and the CPA-resistance. Two alternative, and supposedly stronger, criteria are also presented. Their superiority w.r.t. (9) is still on open issue.

Acknowledgements

This work has partly funded by the French "Conseil Régional de la Région PACA" through the SCS competitivity international pole and by STMicroelectronics Advanced System Technology department at Rousset.

The authors are also grateful to the anonymous reviewers for their valuable comments and suggestions.

Appendix A. The Attacked DES Architecture

The hardware used for the measurements is described in [14]. This section briefly recalls the main features of the hardware.

The DES crypto-processor was willingly embedded within a SoC to avoid interferences between the encryptions and the pads activity: indeed, in the presented architecture, the cryptoprocessor's program is loaded once, and then executes silently; the only pad that toggles is a trigger that is sent to the oscilloscope so that it properly synchronizes the acquisitions. This trigger is activated well before the encryption begins to ensure an optimal decoupling of the two events.

The SecMat V1 experimental circuit is designed to validate countermeasures against the DPA (Differential Power Attacks.) It is made up of about two million transistors and has a silicon area of 4 mm^2. Its overall architecture is a bus-centric system-on-chip (SoC), described in Tab. 1. Standardized modules, implementing the Advanced-VCI [16] interface, are plugged together

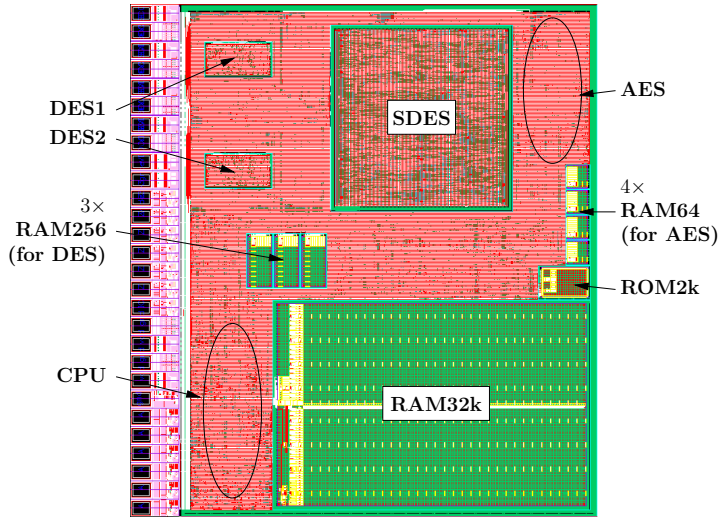


FIGURE 7. Prototype ASIC developed in order to confront power models against actual measurements (refer to [5, pp 62–63].) The target cryptoprocessor is labeled "DES2".

onto a fixed priority bus mastered by an 8-bit 6502 CISC microprocessor (obtained from the late open source project FREE-IP.) The processor boots a "monitor" from an embedded 2kb ROM and loads its program from the outside through an UART (up to 921 600 bauds) into a embedded 32kb RAM. The SoC is programmable in the C language (using `cc65` compiler chain from `http://www.cc65.org/`.) The main feature of the chip is the activation of the four cryptoprocessors — one AES and three DES — to lead DPA campaigns. It has been demonstrated interactively at the circuit exhibition collocated with the conference ESSCIRC'05.

The SecMat circuits were synthesized with Cadence `pks_shell` and placed-and-routed with Cadence `encounter`. The DES modules to cryptanalyze were powered by a dedicated supply pair, that convey the `VSS`=0 volt and `VDD`=1.2 volt voltages directly into to the co-processor, equipped with its own power ring. The private voltage of every co-processor is noted V, whereas the circuit's core voltage is noted U. In normal operating conditions, V=U=1.2 volt. For the sake of attacks, the voltages may be tuned, as shown in the left part of Fig. 8.

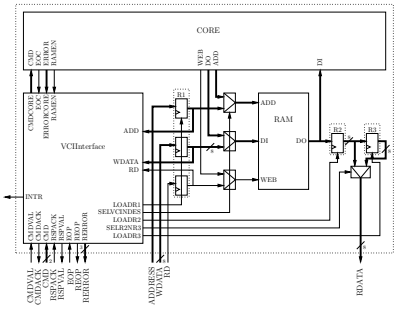
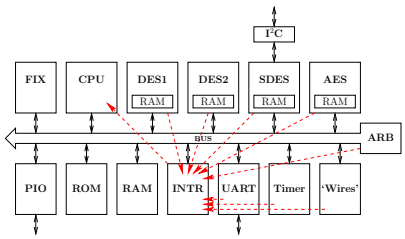
SecMat V1 Modules	SecMat V1 Top-Level
Every module (*e.g.* crypto-processors) communicates via a shared local RAM. 	Modules are connected to a bus, and are able to send interrupts to the 6502 CPU. 

TABLE 1. SecMat V1 System-on-Chip architecture.

The process is a low-leakage (hence high threshold voltage — $V_{\text{th}}(N) = 295$ mV & $V_{\text{th}}(P) = 367$ mV) 130 nm technology with 6 metal layers (M1–M6) from STMicroelectronics. The chips are fabricated through the multi-project wafers offered by the CMP (`http://cmp.imag.fr/`.)

The SecMat V1 circuit is placed on a motherboard that is controllable remotely via a single USB socket. The SECMAT circuit monitor is functional and can execute arbitrary code injected from a PC. The attack motherboard is shown in the right part of Fig. 8.

Appendix B. The Acquisition Setup

The acquisition apparatus is an Infiniium 54 855A oscilloscope sold by Agilent. The probes' model is 1134A, featuring a bandwidth of 7 GHz. The E2669A differential connectivity kit was used. The power traces shown in this document were acquired with a solder-in connector.

This section reports an acquisition campaign realized on the DES hardware encryption of the ASIC SecMat V1. The architecture of the crypto-processor is extensively described in chapter 3 of [13]. The campaign consists in the acquisition of 81 089 traces with a constant key, `jexjexje` in ASCII or `0x6a65786a65786a65`

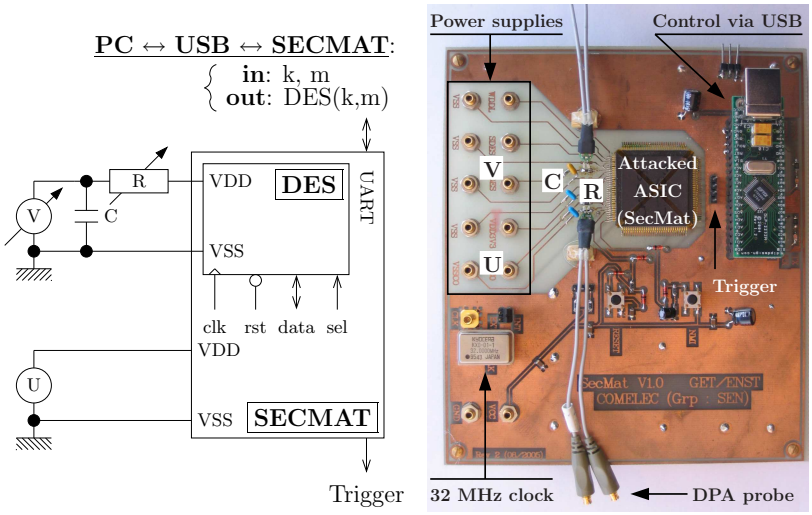


FIGURE 8. Tunable environmental conditions (R, V) when measuring side-channels on the SecMat DES co-processor. (*left.*) The attack board front view, with the SecMat V1 ASIC in exergue. (*right.*)

in hexadecimal. The traces are averaged 64 times by the oscilloscope. Without averaging, the traces resolution is 8 bits. Using the oscilloscope built-in averaging capability, the resolution can reach 12 bits. The spying resistor is on the VDD side of the power supply, its resistance is 11 Ω, and the voltage is the nominal value of 1.2 volts. In Fig. 3, another experimental condition is also used (V = 0.7 volts and R=80 Ω.)

Appendix C. Mathematical Proofs

C.1. Proof of the First Equation in (2)

Proof.

$$
\begin{aligned}
& \mathbb{E}\left(\left\{\sum_{i\in\text{nets}} \begin{array}{l}\xi_i^{\uparrow}\overline{i}(t)\cdot i(t+1)+\\ \xi_i^{\downarrow} i(t)\cdot\overline{i}(t+1)\end{array}\right\}\times\left(-2\cdot(-1)^{j(t+1)}\right)\right)\\
= \;& -2\cdot\mathbb{E}\left(\left\{\sum_{i\in\text{nets}} \begin{array}{l}\xi_i^{\uparrow}\overline{i}(t)\cdot i(t+1)+\\ \xi_i^{\downarrow} i(t)\cdot\overline{i}(t+1)\end{array}\right\}\times(1-2\times j(t+1))\right)\\
= \;& -2\cdot\sum_{i\in\text{nets}}\left\{\begin{array}{l}\xi_i^{\uparrow}\mathbb{E}\left(\overline{i}(t)\cdot i(t+1)\right)+\\ \xi_i^{\downarrow}\mathbb{E}\left(i(t)\cdot\overline{i}(t+1)\right)\end{array}\right\}\\
& -2\cdot\sum_{i\in\text{nets}}(-2)\times\left\{\begin{array}{l}\xi_i^{\uparrow}\mathbb{E}\left(\overline{i}(t)\cdot i(t+1)\cdot j(t+1)\right)+\\ \xi_i^{\downarrow}\mathbb{E}\left(i(t)\cdot\overline{i}(t+1)\cdot j(t+1)\right)\end{array}\right\}\\
= \;& -2\cdot\sum_{i\in\text{nets}}\left\{\begin{array}{ll}\frac{1}{2^2}\xi_i^{\uparrow}+\frac{1}{2^2}\xi_i^{\downarrow}-2\left\{\frac{1}{2^3}\xi_i^{\uparrow}+\frac{1}{2^3}\xi_i^{\downarrow}\right\} & \text{if } i\neq j\,,\\ \frac{1}{2^2}\xi_i^{\uparrow}+\frac{1}{2^2}\xi_i^{\downarrow}-2\left\{\frac{1}{2^2}\xi_i^{\uparrow}+0\times\xi_i^{\downarrow}\right\} & \text{if } i=j\,.\end{array}\right.\\
= \;& -2\cdot\sum_{i\in\text{nets}}\left\{\begin{array}{ll}0 & \text{if } i\neq j\,,\\ -\frac{1}{2^2}\xi_i^{\uparrow}+\frac{1}{2^2}\xi_i^{\downarrow} & \text{if } i=j\,.\end{array}\right.\\
= \;& \tfrac{1}{2}\cdot(\xi_j^{\uparrow}-\xi_j^{\downarrow})\,.
\end{aligned}
$$

□

C.2. Autocorrelation Lemma Proof

The purpose of this subsection is to show that the autocorrelation of a function f is maximal at its origin (*i.e.* in 0.) The lemma to prove can be expressed in the following way:

$$\forall\varepsilon,\ \mathrm{tr}\left(f^2\right)\geq\mathrm{tr}\left(f\cdot f\circ\tau_\varepsilon\right)\,. \tag{11}$$

Proof. Given an arbitrary $\alpha\in\mathbb{R}$, the expression

$$\mathrm{tr}\left(\alpha\cdot f-f\circ\tau_\varepsilon\right)^2\in\mathbb{R}$$

is trivially greater or equal to zero. Put differently, the $\mathbb{R}\to\mathbb{R}$ application:

$$
\begin{aligned}
\alpha \;&\mapsto\; \mathrm{tr}\left(\alpha\cdot f-f\circ\tau_\varepsilon\right)^2\\
&\mapsto\; \mathrm{tr}\left(\alpha\cdot f\right)^2+\mathrm{tr}\left(f\circ\tau_\varepsilon\right)^2-2\cdot\mathrm{tr}\left(\alpha\cdot f\cdot f\circ\tau_\varepsilon\right)\\
&\mapsto\; \alpha^2\cdot\underbrace{\mathrm{tr}f^2}_{a}\;-\;\alpha\cdot\underbrace{2\cdot\mathrm{tr}\left(f\cdot f\circ\tau_\varepsilon\right)}_{b}\;+\;\underbrace{\mathrm{tr}f^2}_{c}
\end{aligned}
$$

is positive or null. As it is a parabola, it has either a double zero or no real root at all. The quadratic discriminant $b^2 - 4 \cdot a \cdot c$ is thus either null or strictly negative, *i.e.* $b^2 \leq 4 \cdot a \cdot c$. Hence:

$$(2 \cdot \mathrm{tr}\,(f \cdot f \circ \tau_\varepsilon))^2 \leq 4\big(\mathrm{tr} f^2\big)^2 .$$

Given that $\mathrm{tr} f^2$ is positive, the square root of the previous inequality can be taken safely:

$$\mathrm{tr} f^2 \geq \pm \mathrm{tr}\,(f \cdot f \circ \tau_\varepsilon) .$$

This is sufficient to prove the announced lemma. □

Another way of proving Eqn. (11) consists in using linear algebra results, as explained below:

Proof. The set of real-valued functions $\mathcal{E} = (\mathbb{F}_2^n \to \mathbb{R}, +, \cdot)$ is a vectorial space on $\mathbb{R}$. It is casually referred to as "pseudo-Boolean" functions [1]. The application $(f, g) \mapsto \langle f, g \rangle \doteq \mathrm{tr}(f \cdot g)$ is a scalar product on $\mathcal{E}$, because it is a symmetric positive-definite bilinear form. Eqn. (11) is thus a mere special case of the Cauchy-Schwarz theorem with $g = f \circ \tau_\varepsilon$. □

References

[1] E. Boros and P.L. Hammer. Pseudo-Boolean Optimization. *Discrete Applied Mathematics*, 123((1-3)):155–225, 2002.

[2] Cadence. Delay Calculation Algorithm Guide, june 2002. Product SPR50, `ct_alg.pdf`.

[3] Claude Carlet. On Highly Nonlinear S-Boxes and Their Inability to Thwart DPA Attacks. pages 49–62. INDOCRYPT 2005 (LNCS 3797), december 2005. Bangalore, India. (Complete version on IACR ePrint).

[4] S. Chari, J.R. Rao, and P. Rohatgi. Template Attacks. In *CHES*, volume 2523 of *LNCS*, August 2002. ISBN: 3-540-00409-2.

[5] "Circuits Multi-Projets" (CMP, < cmp@imag.fr >) Annual Report 2005.

[6] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis: Leaking Secrets. In *Proceedings of CRYPTO'99*, volume 1666 of *LNCS*, pages pp 388–397. Springer, 1999.

[7] Neil H.E. Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. 3 edition (May 11, 2004), ISBN: 0321149017.

[8] NIST/ITL/CSD. Data Encryption Standard. FIPS PUB 46-3, Oct 1999. `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`.

[9] Paul N. Fahn and Peter K. Pearson. IPA: A New Class of Power Attacks. 1717/1999:173, August 1999. Worcester, MA, USA. ISSN 0302-9743.

[10] Emmanuel Prouff. DPA Attacks and S-Boxes. pages 424–441. FSE 2005 (LNCS 3557), february 2005. Paris, France. (Edited by Springer-Verlag).

[11] Régis Bévan. *Évaluation statistique et sécurité des cartes à puce. Évaluation d'attaques DPA évoluées.* PhD thesis, (french). Université Paris 11 & École Nationale Supérieure d'Électricité (Supélec), April 2004.

[12] S. Guilley and Ph. Hoogvorst and R. Pacalet. Differential Power Analysis Model and some Results. In *Proceedings of WCC/CARDIS'04*, pages pp 127–142, August 2004. Toulouse, France.

[13] Sylvain Guilley. *Contre-mesures Géométriques aux Attaques Exploitant les Canaux Cachés*. PhD thesis, ENST, January 2007.

[14] Sylvain Guilley and Philippe Hoogvorst and Renaud Pacalet. A Fast Pipelined Multi-Mode DES Architecture Operating in IP Representation. *Integration, The VLSI Journal*,
DOI: 10.1016/j.vlsi.2006.06.004. (To appear in 2007).

[15] Thomas S. Messerges and Ezzy A. Dabbish and Robert H. Sloan. Investigations of Power Analysis Attacks on Smartcards. In *USENIX — Smartcard'99*, pages 151–162, May 10–11 1999. Chicago, Illinois, USA.

[16] VSI Alliance. On-Chip Bus Development Working Group. Virtual Component Interface Standard Version 2 (OCB 2 2.0), April 2001.
`http://www.vsia.org/`.

[17] Éric Brier, Christophe Clavier, and Francis Olivier. Optimal statistical power analysis. Cryptology ePrint Archive, Report 2003/152, 2003.

[18] Éric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. *Proc. of CHES'04*, LNCS 3156:16–29, August 11–13 2004. ISSN: 0302-9743; ISBN: 3-540-22666-4; DOI: 10.1007/b99451; Cambridge, MA, USA.

SOME RESULTS ON NONLINEARITY PRESERVING BIJECTIVE TRANSFORMATIONS

İsa Sertkaya[1] and Ali Doğanaksoy[1]

Abstract. Nonlinearity is the most crucial criterion for measuring the confusion property of symmetric ciphers. Meier and Staffelbach investigated under which transformations nonlinearity is preserved by only considering the bijective transformations that act on input arguments of Boolean functions. In this study, we extend the group by including the bijective transformations acting on the truth table of Boolean functions. We first give necessary and sufficient conditions for an affine bijective transformation to keep nonlinearity invariant. Then, we present a class of non-affine nonlinearity preserving bijective transformations and with some search results, we give exact set of nonlinearity preserving bijective transformations for Boolean functions with two input arguments.
Keywords: Boolean function, nonlinearity

1. Introduction

Confusion is a very important property for a symmetric cipher, lack of which causes vulnerability to cryptanalytic attacks. In order to resist these attacks, the building components of the cipher, mostly viewed as Boolean functions, are chosen with high nonlinearity.

Investigating the bijective transformations that preserve the design criterion will not only help understanding how powerful

[1] Institute of Applied Mathematics, Middle East Technical University and National Research Institute of Electronics and Cryptology, TÜBİTAK-UEKAE, Turkey.
email: isa@uekae.tubitak.gov.tr, aldoks@metu.edu.tr

the criterion is [1], but may also give opportunity to classify the Boolean functions with respect to the criterion or to construct Boolean functions with desired confusion and diffusion properties.

Meier and Staffelbach [4] studied and classified the bijective transformations acting on input arguments of Boolean functions. They showed that, only the affine transformations keep nonlinearity invariant. However, the considered group is a small group related to all bijective transformations that act on the truth tables of Boolean functions.

Later, in [3] (page 417) and [5] (Proposition 8.3), some specific bijective transformations are also investigated related to the Walsh spectrum of Boolean functions. The considered transformations are composed of an affine bijective transformation on input arguments of Boolean functions and adding an affine Boolean functions. These results are a special case of the Theorem 3.7 stated in the following section.

In this study, we concentrate on the group of the bijective transformations and try to determine those transformations that keep nonlinearity invariant. After recalling the result obtained in [4], we give necessary and sufficient conditions for an affine bijective transformation to preserve nonlinearity. Later, we focus on non-affine bijective transformations and show existence of such transformations. Finally, by using the search results, we give the exact set of nolinearity preserving bijective transformations for Boolean function with two input arguments and give some examples of non-affine nolinearity preserving bijective transformations for Boolean function with two input arguments.

2. Preliminaries

In this section, we fix the notation and state the definitions. A *Boolean function* on n variables is a map, with domain $\mathcal{V}_n$, that takes values from $GF(2)$. The set of all Boolean functions is denoted by $\mathcal{F}_n$, which is isomorphic to $\mathcal{V}_{2^n}$, and trivially $|\mathcal{F}_n| = 2^{2^n}$. From now on, unless otherwise stated explicitly, by "a function" we mean a Boolean function in $\mathcal{F}_n$.

Any function f can be uniquely represented as;

- The *truth table* of f,

$$T_f = (f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{2^n-1}))$$

where $\alpha_i \in \mathcal{V}_n$ and α_i's are written in lexicographic order,

- The *sequence* of f,

$$\zeta_f = ((-1)^{f(\alpha_0)}, (-1)^{f(\alpha_1)}, \ldots, (-1)^{f(\alpha_{2^n-1})}),$$

- The *algebraic normal form* of f,

$$f(x_1, x_2, \ldots, x_n) = c_0 \oplus c_1 x_1 \oplus \cdots \oplus c_{12} x_1 x_2 \oplus \cdots \oplus c_{12\cdots n} x_1 x_2 \cdots x_n$$

where $c_0, c_1, \ldots, c_{12\cdots n} \in GF(2)$, or equivalently,

$$ANF_f = (c_0, c_1, \ldots, c_{12\cdots n}).$$

A function is called *affine* if it is of the form

$$f(x_1, x_2, \ldots, x_n) = c_0 \oplus c_1 x_1 \oplus \cdots \oplus c_n x_n.$$

The set of all affine functions is denoted by $\mathcal{A}_n$. If $c_0 = 0$, then the function is called *linear* and the set of all linear functions is denoted by $\mathcal{L}_n$.

The *Walsh transform* of f is defined as:

$$W_f(w) = \sum_{x \in \mathcal{V}^n} (-1)^{f(x) \oplus wx}$$

where $w \in \mathcal{V}_n$ and wx is the standard inner product on $\mathcal{V}_n$. The ordered tuple, $(W_f(\alpha_0), W_f(\alpha_1), \ldots, W_f(\alpha_{2^n-1}))$ is called the *Walsh spectrum* of f and is denoted by T_{W_f}. Obviously,

$$T_{W_f} = \zeta_f H_n,$$

where H_n is the *Sylvester-Hadamard matrix* of order 2^n.

The absolute maximum value in the Walsh spectrum is called the *spectral amplitude* and we shall denote it by $\|T_{W_f}\|$.

Nonlinearity N_f of f, is the minimum distance of f to affine functions, that is:

$$N_f = 2^{n-1} - \frac{1}{2} \|T_{W_f}\|$$

Let $\mathcal{S}_n$ be the group consisting of all permutation matrices of order n, and let $\mathcal{D}_n$ be the group consisting of all diagonal matrices with all diagonal entries equal to ± 1. Then $\mathcal{S}_n^{\pm}$, the group generated by $\mathcal{S}_n$ and $\mathcal{D}_n$, is a semidirect product of $\mathcal{S}_n$ and $\mathcal{D}_n$. The elements of $\mathcal{S}_n^{\pm}$ are called *monomial matrices*.

Definition 2.1. [2] Two Hadamard matrices H_1 and H_2 of order n, are *equivalent* if one can be obtained from the other by operations of permuting rows and multiplying some rows by -1, and/or permuting columns and multiplying some columns by -1. That is to say, H_1 and H_2 are equivalent if $H_2 = P^{-1}H_1Q$, for some monomial matrices P, Q in $\mathcal{S}_n^{\pm}$.

Definition 2.2. [2] The *automorphism group* of a Hadamard matrix H of order n, is the group consisting of all pairs (P, Q) of monomial matrices, satisfying $P^{-1}HQ = H$, where the group operation is $(P_1, Q_1) \circ (P_2, Q_2) = (P_1P_2, Q_1Q_2)$.

We denote the automorphism group of a Hadamard matrix H by $Aut(H)$, that is

$$Aut(H) = \{(P, Q) \in \mathcal{S}_n^{\pm} | P^{-1}HQ = H\}.$$

We end this section by stating some theorems, that will be used in the following section, without giving proof. For further deatails, reader may refer to [6].

Theorem 2.3. *[6] Let $f, g \in \mathcal{F}_n$. Let $h \in \mathcal{F}_n$ be the function defined as,*

$$h(\alpha) = (f \oplus g)(\alpha) = f(\alpha) \oplus g(\alpha) \text{ for all } \alpha \in \mathcal{V}_n.$$

Then, the Walsh transform of h is equal to

$$W_h(\omega) = \frac{1}{2^n} \sum_{\alpha \in \mathcal{V}_n} W_f(\omega \oplus \alpha) W_g(\alpha),$$

for all $\omega \in \mathcal{V}_n$.

Corollary 2.4. *([6]) Let $f \in \mathcal{F}_n$ and $g \in \mathcal{L}_n$ be such that $g(\alpha) = \beta\alpha$ for all $\alpha \in \mathcal{V}_n$. Then, the Walsh transform of the function $h \in \mathcal{F}_n$ defined as,*

$$h(\alpha) = (f \oplus g)(\alpha) = f(\alpha) \oplus \beta\alpha \text{ for all } \alpha \in \mathcal{V}_n,$$

is equal to

$$W_h(\omega) = W_f(\omega \oplus \beta),$$

for all $\omega \in \mathcal{V}_n$.

3. Nonlinearity Preserving Bijective Transformations

Let $\Theta(n)$ be the group of bijective transformations over V_{2^n}. Any transformation $\psi \in \Theta(n)$, can be written as:

$$\begin{aligned}\psi(x_1, x_2, \ldots, x_{2^n}) &= (\psi_0(x_1, x_2, \ldots, x_{2^n}), \psi_1(x_1, x_2, \ldots, x_{2^n}),\\ &\quad \ldots, \psi_{2^n-1}(x_1, x_2, \ldots, x_{2^n})),\end{aligned}$$

where $\psi_i(x_1, x_2, \ldots, x_{2^n}) \in \mathcal{F}_{2^n}$, $i = 0, 1, \ldots, 2^n-1$. We denote the action of a transformation $\psi \in \Theta(n)$ on a function $f \in \mathcal{F}_n$ by $\psi * f = \psi(T_f)$. Note that $(\psi * f)(\alpha_i) = \psi_i(T_f)$, $i \in \{0, 1, \ldots, 2^n - 1\}$. Thus, $\psi * f$ is the truth table of the resulting function, and $(\psi * f)(\alpha_i)$ is the i-th $(i = 0, 1, \ldots, 2^n - 1)$ component of the truth table, namely the image of the resulting function at $\alpha_i \in \mathcal{V}_n$.

It is obvious that, a $\psi \in \Theta(n)$ preserves nonlinearity, that is $N_f = N_{\psi * f}$, if and only if $\left\|T_{W_f}\right\| = \left\|T_{W_{\psi * f}}\right\|$, for any $f \in \mathcal{F}_n$. By $\mathcal{P}_n(N)$, we denote the set of transformations, which preserve N_f for all $f \in \mathcal{F}_n$, i.e.,

$$\mathcal{P}_n(N) = \{\psi \in \Theta(n) \mid N_f = N_{\psi * f}, \text{ for all } f \in \mathcal{F}_n\}$$

Theorem 3.1. *Let $\psi \in \Theta(n)$, such that ψ results in a permutation in the truth table for all $f \in \mathcal{F}_n$, that is $\psi * f = T_f P$ where $P \in \mathcal{S}_{2^n}$. Then, $\psi \in \mathcal{P}_n(N)$ if and only if there exists a $Q \in \mathcal{S}_{2^n}^{\pm}$ such that $(Q^{-1}, P) \in Aut(H_n)$.*

Proof. Obviously, if there exists a $Q \in \mathcal{S}_{2^n}^{\pm}$ such that $(Q^{-1}, P) \in Aut(H_n)$, then $T_{W_{\psi * f}}$ is nothing but a signed permutation of T_{W_f}. Thus, since spectral amplitude does not change, we can conclude that $\psi \in \mathcal{P}_n(N)$.

Note that, $\psi * f = T_f P$ means ψ is a bijective transformation on input arguments of functions. Thus, by [4] we know that, ψ is in fact an affine bijective transformation on input arguments of f, i.e., $\psi * f = T_f P = T_h$ where

$$h(\alpha) = f(\alpha A \oplus \beta) \text{ for all } \alpha \in \mathcal{V}_n,$$

where $\beta \in \mathcal{V}_n$ and $A \in GL(n, GF(2))$. Then, for all $f \in \mathcal{F}_n$ we have,

$$\begin{aligned} W_h(\omega) &= \sum_{\alpha \in \mathcal{V}_n} (-1)^{f(\alpha A \oplus \beta) \oplus \omega\alpha} \\ &= \sum_{\alpha \in \mathcal{V}_n} (-1)^{f(\alpha) \oplus \omega(\alpha A^{-1} \oplus \beta A^{-1})} \\ &= \sum_{\alpha \in \mathcal{V}_n} (-1)^{f(\alpha) \oplus (\omega(\alpha A^{-1}) \oplus \omega(\beta A^{-1}))} \\ &= (-1)^{\omega(\beta A^{-1})} \sum_{\alpha \in \mathcal{V}_n} (-1)^{f(\alpha) \oplus (\omega(A^{-1})^t)\alpha}, \end{aligned}$$

for all $\omega \in \mathcal{V}_n$.

Thus, $\psi \in \mathcal{P}_n(N)$ if and only if there exists a $Q \in \mathcal{S}_{2^n}^{\pm}$ such that $(Q^{-1}, P) \in Aut(H_n)$. □

Theorem 3.1 restates the result of Meier and Staffelbach, by viewing the transformations as acting on the truth table of the functions. Furthermore, with the following corollary, it classifies the affine bijective transformations whether they are in fact linear or not.

Corollary 3.2. *Given a permutation matrix $P \in \mathcal{S}_{2^n}$, the corresponding bijective transformation that acts on input arguments of f is linear if and only if $Q \in \mathcal{S}_{2^n}$, where $(Q^{-1}, P) \in Aut(H_n)$.*

Proof. Immediately follows from the proof of Theorem 3.1. □

Naturally, $\Theta(n)$ can be partitioned as affine and non-affine (nonlinear) transformations. Recall that by definition, $\psi \in \Theta(n)$ can be written as:

$$\begin{aligned} \psi(x_1, x_2, \ldots, x_{2^n}) &= (\psi_0(x_1, x_2, \ldots, x_{2^n}), \psi_1(x_1, x_2, \ldots, x_{2^n}), \\ &\quad \ldots, \psi_{2^n-1}(x_1, x_2, \ldots, x_{2^n})), \end{aligned}$$

where $\psi_i(x_1, x_2, \ldots, x_{2^n}) \in \mathcal{F}_{2^n}$, $i = 0, 1, \ldots, 2^n - 1$. Since, each $\psi_i \in \mathcal{F}_{2^n}$ can be represented by the algebraic normal form:

$$\psi_i(x_1, x_2, \cdots, x_{2^n}) = c_0^{(i)} \oplus c_1^{(i)} x_1 \oplus \ldots \oplus c_{12\ldots 2^n}^{(i)} x_1 x_2 \cdots x_{2^n}.$$

Then we get,

$$\psi : T_f \longmapsto \Bigg(\underbrace{\begin{bmatrix} c_0^{(0)} \\ c_0^{(1)} \\ \vdots \\ c_0^{(2^n-1)} \end{bmatrix}}_{\lambda_0} \oplus \underbrace{\begin{bmatrix} c_1^{(0)} \\ c_1^{(1)} \\ \vdots \\ c_1^{(2^n-1)} \end{bmatrix}}_{\lambda_1} f(\alpha_0) \oplus \cdots \oplus$$

$$\underbrace{\begin{bmatrix} c_{2^n}^{(0)} \\ c_{2^n}^{(1)} \\ \vdots \\ c_{2^n}^{(2^n-1)} \end{bmatrix}}_{\lambda_{2^n}} f(\alpha_{2^n-1}) \oplus \underbrace{\begin{bmatrix} c_{12}^{(0)} \\ c_{12}^{(1)} \\ \vdots \\ c_{12}^{(2^n-1)} \end{bmatrix}}_{\lambda_{12}} f(\alpha_0) f(\alpha_1) \oplus$$

$$\cdots \oplus \underbrace{\begin{bmatrix} c_{12\cdots 2^n}^{(0)} \\ c_{12\cdots 2^n}^{(1)} \\ \vdots \\ c_{12\cdots 2^n}^{(2^n-1)} \end{bmatrix}}_{\lambda_{12\cdots 2^n}} f(\alpha_0) \cdots f(\alpha_{2^n-1}) \Bigg)^t ,$$

or equivalently,

$$\psi : T_f \longmapsto \big(\lambda_0 \oplus AT_f^t \oplus \lambda_{12} f(\alpha_0) f(\alpha_1) \oplus \cdots \oplus \lambda_{12\cdots 2^n} f(\alpha_0) f(\alpha_1) \cdots f(\alpha_{2^n-1})\big)^t ,$$

where A is the matrix of the form $A = [\lambda_1\ \lambda_2\ \ldots\ \lambda_{2^n}]$.

For a bijective transformation $\psi \in \Theta(n)$, if $\lambda_i = [0\ 0\ \ldots\ 0]^t$ for all $i \in \{12, 13, \ldots, 12\cdots 2^n\}$, then ψ is an affine transformation. Otherwise, if there exists a $\lambda_i \neq [0\ 0\ \ldots\ 0]^t$ for some $i \in \{12, 13, \ldots, 12\cdots 2^n\}$, then ψ is a non-affine transformation. We denote the subgroup of $\Theta(n)$ consisting of all affine transformations by $\mathcal{A}(n)$.

The bijective transformations acting on input arguments of the function $f \in \mathcal{F}_n$ are the elements of $\Theta(n)$, that satisfies the following conditions:

i: $\lambda_0 = [0\ 0\ \ldots\ 0]^t$,

ii: $\lambda_i = [0\ 0\ \ldots\ 0]^t$, for all $i \in \{12, 13, \ldots, 12\cdots 2^n\}$,
iii: $A \in \mathcal{S}_{2^n}$, i.e., A is a permutation matrix.

Hence, the group of all bijective transformations acting on input arguments of the function f, is in fact the group $\mathcal{S}_{2^n}$. Therefore, we can conclude that $\mathcal{S}_{2^n} \subset \mathcal{A}(n) \subset \Theta(n)$.

The following lemma shows the existence of transformations other than permutations which keep N_f invariant for all $f \in \mathcal{F}_n$.

Lemma 3.3. *Fix $g \in \mathcal{F}_n$, and define a transformation $\psi \in \Theta(n)$ by setting $\psi * f = \psi(T_f) = T_f \oplus T_g$, for all $f \in \mathcal{F}_n$. Then, $\psi \in \mathcal{P}_n(N)$ if and only if $g \in \mathcal{A}_n$.*

Proof. First, suppose $\psi \in \Theta(n)$ such that $\psi(T_f) = T_f \oplus T_g$ where $g \in \mathcal{A}_n$, and let $T_h = \psi * f = T_f \oplus T_g$. Then, from Theorem 2.3, we know that, the Walsh transform of the function h is

$$W_h(\omega) = \frac{1}{2^n} \sum_{\alpha \in \mathcal{V}_n} W_f(\omega \oplus \alpha) W_g(\alpha),$$

for all $\omega \in \mathcal{V}_n$. Since $g \in \mathcal{A}_n$, $T_{W_{\hat{g}}} = (0, 0, \ldots, \pm 2^n, 0, \ldots, 0)$. That is, Walsh spectrum of g has only one non-zero value, which is equal to $\pm 2^n$. Then, the above equation can be simplified as,

$$W_h(\omega) = \pm W_f(\omega \oplus \alpha_k)$$

for all $\omega \in \mathcal{V}_n$, where $k \in \{0, 1, \ldots, 2^n - 1\}$. Thus, ψ preserves $\left\|T_{W_f}\right\|$ for all $f \in \mathcal{F}_n$. Therefore, it follows that $\psi \in \mathcal{P}_n(N)$.
Now conversely, suppose $\psi \in \Theta(n)$ such that $\psi(T_f) = T_f \oplus T_g$ where $f, g \in \mathcal{F}_n$ and $\psi \in \mathcal{P}_n(N)$. Assume that $g \notin \mathcal{A}_n$. Consider a function $f \in \mathcal{A}_n$. Then, obviously, the function $h \in \mathcal{F}_n$, with $T_h = \psi * f = T_f \oplus T_g$ is not an affine function. Thus, $N_h > N_f$, which contradicts with the assumption $\psi \in \mathcal{P}_n(N)$. Thus, the assertion follows. □

Now, we give some lemmas that will be used in the proof of the main theorem which states the necessary and sufficient conditions for an affine transformation $\psi \in \mathcal{A}(n)$ to preserve nonlinearity for all $f \in \mathcal{F}_n$. For the proofs, reader may refer to [6] and [7].

Lemma 3.4. *[6] Let $A \in GL(2^n, GF(2))$ be fixed. Define $\psi \in \Theta(n)$ such that $\psi(T_f) = T_f A$ for all $f \in \mathcal{F}_n$. If $\psi \in \mathcal{P}_n(N)$, then $A = (a_{i,j})$, where $i, j \in \{1, 2, \ldots, 2^n\}$, satisfies the following:*

i: *Any column of A is the truth table of some function whose nonlinearity is equal to one.*

ii: *The vector obtained by xor of any two columns of A is the truth table of some function whose nonlinearity is equal to two.*

Lemma 3.5. *[6] Any matrix $A \in GL(2^n, GF(2))$ satisfies the two conditions in Lemma 3.4 if and only if $A = B \oplus P$, where $P \in \mathcal{S}_{2^n}$ and B is a matrix of order 2^n over $GF(2)$ whose columns are the not necessarily distinct truth table of affine functions.*

Lemma 3.6. *[6] Given $A \in GL(2^n, GF(2))$ satisfying the conditions in Lemma 3.4, define the linear transformation $\psi \in \Theta(n)$ so that $\psi(T_f) = T_f A$ for all $f \in \mathcal{F}_n$. Then, $\psi \in \mathcal{P}_n(N)$ if and only if the corresponding $P \in \mathcal{S}_{2^n}$ is in fact an affine transformation acting on input arguments of f.*

Lemma 3.6 states that the nonlinearity preserving linear transformations are of the form $T_f(B \oplus P)$ where $P \in \mathcal{S}_{2^n}$ corresponds to an affine transformation on input arguments of f, and B is a matrix of order 2^n over $GF(2)$ whose columns are truth tables of some affine functions, not necessarily distinct. Let ψ be a nonlinearity preserving linear transformation. Trivially, for all $f \in \mathcal{F}_n$ we have $\psi * f = T_f(B \oplus P)$ where B and P are as mentioned above. Furthermore, we can express ψ as $\psi = \psi_1 \oplus \psi_2$ where $\psi_1 * f = T_f B$ and $\psi_2 * f = T_f P$. Obviously, image of ψ_1 is a subset of affine functions, i.e., $Im(\psi_1) \subseteq \mathcal{A}_n$. So, we may replace ψ_1 with a nonlinear map defined over V_{2^n}, say $\tilde{\psi}_1$, with $Im(\tilde{\psi}_1) \subseteq \mathcal{A}_n$, which is not necessarily invertible. Then, the resulting transformation $\tilde{\psi} = \tilde{\psi}_1 \oplus \psi_2$ is not an affine transformation. However, it is easy to show that $\tilde{\psi}$ keeps nonlinearity invariant. Therefore, if there exists a bijective transformation $\tilde{\psi}$ such that $\tilde{\psi} = \tilde{\psi}_1 \oplus \psi_2$ where $Im(\tilde{\psi}_1) \subseteq \mathcal{A}_n$ and $\psi_2 * f = T_f P$ with P satifying the requirements above, then $\tilde{\psi}$ is a non-affine nonlinearity preserving transformation.

Theorem 3.7. *Let $\psi \in \mathcal{A}(n)$ be an affine transformation so that for all $f \in \mathcal{F}_n$,*

$$\psi * f = \psi(T_f) = T_g \oplus T_f A,$$

where $g \in \mathcal{F}_n$ and $A \in GL(2^n, GF(2))$ are fixed.

Then, $\psi \in \mathcal{P}_n(N)$ if and only if $g \in \mathcal{A}_n$ and $A = B \oplus P$, where $P \in \mathcal{S}_{2^n}$ corresponds to an affine transformation acting on input arguments of f, and B is the matrix of order 2^n over $GF(2)$ whose

columns are the truth table of affine functions, not necessarily distinct.

Proof. For fixed $g \in \mathcal{F}_n$ and $A \in GL(2^n, GF(2))$, suppose $\psi \in \Theta(n)$ so that $\psi(T_f) = T_g \oplus T_f A$, preserves N_f for all $f \in \mathcal{F}_n$, that is $\psi \in \mathcal{P}_n(N)$. Now, consider the affine function 0_n, that is $T_{0_n} = (0, 0, \ldots, 0)$. Then, by the assumption, the function $h \in \mathcal{F}_n$ such that $T_h = \psi(T_{0_n})$ has the same nonlinearity with 0_n. In other words, $h \in \mathcal{A}_n$. Since, $T_h = \psi(T_{0_n}) = T_g$, it follows that $g \in \mathcal{A}_n$. Since ψ is an affine transformation, we can write $\psi * f = \psi_2 * (\psi_1 * f)$, where $\psi_1 * f = T_f A$ and $\psi_2 * f = T_f \oplus T_g$. We have shown that, ψ_2 preserves N_f, for all $f \in \mathcal{F}_n$, since $g \in \mathcal{A}_n$. Then, $\psi \in \mathcal{P}_n(N)$ if and only if ψ_1 preserves N_f, for all $f \in \mathcal{F}_n$, for which the sufficient and necessary conditions are shown in Lemma 3.5 and 3.6.
Obviously, from Lemma 3.3, 3.4, 3.5, and 3.6, it follows that if $g \in \mathcal{A}_n$ and $A = B \oplus P$, where $P \in \mathcal{S}_{2^n}$ with $\varphi \in \mathcal{A}(n)$, and B is the matrix of order 2^n over $GF(2)$, whose columns are the truth table of affine functions, which are not necessarily distinct, and $\psi(T_f) = T_g \oplus T_f A$, then $\psi \in \mathcal{P}_n(N)$. □

Note that, the results obtained in [3] and [5] are in fact, equivalent to a special case of the above theorem, that is the case when the matrix B is the zero matrix.

So far, we have analyzed the elements of $\mathcal{A}(n)$ and classified whether they are in $\mathcal{P}_n(N)$ or not, by giving the necessary and sufficient conditions on affine bijective transformations to keep nonlinearity invariant. Now, we present a proposition that shows the existence of non-affine bijective transformations that keep nonlinearity invariant.

Proposition 3.8. *Let $\psi \in \Theta(n)$ be a transformation satisfying the following conditions:*

- *i*: *λ_0 is the truth table of some affine Boolean function,*
- *ii*: *the matrix A satisfies the conditions mentioned in Lemma 3.6,*
- *iii*: *λ_i is the truth table of some affine Boolean function for all $i \in \{12, 13, \ldots, 12 \cdots 2^n\}$.*

Unless $\lambda_i = [0\ 0\ \ldots\ 0]^t$ for all $i \in \{12, 13, \ldots, 12 \cdots 2^n\}$, ψ is a non-affine transformation. Then, $\psi \in \Theta(n)$ satisfying the above conditions preserve nonlinearity for all $f \in \mathcal{F}_n$.

Proof. The action of $\psi \in \Theta(n)$ satisfying the mentioned conditions, on a function is nothing but permuting the function's truth table by an affine transformation that acts on input arguments, and xoring it with the truth table of an affine function which is determined by the function itself. We know by Theorem 3.1 and Lemma 3.3, such a situation does not change the nonlinearity for all $f \in \mathcal{F}_n$. Hence, all $\psi \in \Theta(n)$ that satisfy the above conditions, surely, keep nonlinearity invariant. □

Proposition 3.8 proves that there exists a class of non-affine nonlinearity preserving bijective transformations. In order to make the proposed conditions on non-affine bijective transformations clear, we give some examples on $\mathcal{F}_2$ in the following section.

4. Search Results and Examples

The number of elements in $\Theta(n)$ is $2^{2^n}!$. In fact, for $n = 2$, we have:

(1) $|\mathcal{F}_2| = 16$
(2) $|\mathcal{A}_2| = 8$
(3) $|\Theta(2)| = 16! \cong 2^{44}$
(4) $|\mathcal{P}_2(N)| = 8! \times 8! \cong 2^{30}$

where $|.|$ stands for the cardinality of that set, i.e. number of elements of the set. Here, $\mathcal{P}_2(N)$ can be easily constructed since, in $\mathcal{F}_2$, there exists 8 affine functions and constructing a transformation in a way that maps affine functions onto itself and non-affine functions onto non-affine functions.

Search algorithms can be carried over these bijective transformations when $n = 2$, for extracting the exact set of nonlinearity preserving transformations. But, even for $n = 3$, $|\Theta(3)| \cong 2^{1684}$, hence searching the whole set becomes infeasible.

For $n = 2$, we scan the group of bijective tranformations acting on $\mathcal{F}_2$, namely $\Theta(2)$, in order to clearify the exact set of nonlinearity preserving transformations, namely $\mathcal{P}_2(N)$. The search was pursued by first constructing non-linearity preserving transformations and then checking whether they satisfy the conditions mentioned in Proposition 3.8 or not. Relying on our search results, we prove the following remark:

Proposition 4.1. *Let $\psi \in \Theta(2)$. Then, ψ preserves nonlinearity, i.e. $\psi \in \mathcal{P}_2(N)$, if and only if ψ satisfies the following conditions:*

i: λ_0 *is the truth table of some affine Boolean function,*
ii: *the matrix A satisfies the conditions mentioned in Lemma 3.6,*
iii: λ_i *is the truth table of some affine Boolean function for all* $i \in \{12, 13, \ldots, 1234\}$.

Hence, Propositions 3.8 and 4.1 derives the exact nonlinearity preserving transformations in $\Theta(2)$.

We end this section by giving some examples non-affine nonlinearity preserving transformations for $n = 2$.

Example 4.2. Let $\psi \in \Theta(2)$ defined as:

$$\begin{aligned}\psi(x_1, x_2, x_3, x_4) &= (\psi_0(x_1, x_2, x_3, x_4), \psi_1(x_1, x_2, x_3, x_4),\\ &\quad \psi_2(x_1, x_2, x_3, x_4), \psi_3(x_1, x_2, x_3, x_4)),\end{aligned}$$

where the truth table of $\psi_0, \psi_1, \psi_2, \psi_3$ from F_4 is as follows:

$$\begin{aligned}T_{\psi_0} &= (0,1,0,1,0,1,0,1,0,0,0,0,1,1,1,1),\\ T_{\psi_1} &= (0,0,1,1,0,0,1,1,0,1,0,1,1,1,0,0),\\ T_{\psi_2} &= (0,0,0,0,1,1,1,1,0,0,1,1,1,0,1,0),\\ T_{\psi_3} &= (0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1).\end{aligned}$$

Then, the ANF of $\psi_0, \psi_1, \psi_2, \psi_3$ will be:

$$\begin{aligned}ANF_{\psi_0} &= (0,1,0,0,0,0,0,0,0,1,0,0,1,0,0,0),\\ ANF_{\psi_1} &= (0,0,1,0,0,0,0,0,0,1,1,0,1,1,1,0),\\ ANF_{\psi_2} &= (0,0,0,0,1,0,0,0,0,0,1,0,0,1,1,0),\\ ANF_{\psi_3} &= (0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0).\end{aligned}$$

or equivalently,

$$\begin{aligned}\psi_0(x_1, x_2, x_3, x_4) &= x_1 \oplus x_1x_4 \oplus x_3x_4,\\ \psi_1(x_1, x_2, x_3, x_4) &= x_2 \oplus x_1x_4 \oplus x_2x_4 \oplus x_3x_4 \oplus x_1x_3x_4\\ &\quad \oplus x_2x_3x_4,\\ \psi_2(x_1, x_2, x_3, x_4) &= x_3 \oplus x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4,\\ \psi_3(x_1, x_2, x_3, x_4) &= x_4.\end{aligned}$$

Then we have,

$$\begin{aligned}\psi : T_f \quad \longmapsto \quad & (AT_f^t \oplus \lambda_{14} f(\alpha_0) f(\alpha_3) \oplus \lambda_{24} f(\alpha_1) f(\alpha_3) \oplus \\ & \lambda_{34} f(\alpha_2) f(\alpha_3) \oplus \lambda_{134} f(\alpha_0) f(\alpha_2) f(\alpha_3) \oplus \\ & \lambda_{234} f(\alpha_1) f(\alpha_2) f(\alpha_3))^t,\end{aligned}$$

where A is the identity matrix, $\lambda_{14} = [1100]^t$, $\lambda_{24} = [0110]^t$, $\lambda_{34} = [1100]^t$, $\lambda_{134} = [0110]^t$ and $\lambda_{234} = [0110]^t$.

Obviously, $\psi \notin \mathcal{A}(2)$, that is to say, ψ is not an affine transformation. However, applying ψ to a function is nothing but adding some affine functions to the truth table of the function which are determined by the function itself. Hence, by Lemma 3.3, this does not affect the nonlinearity. Since, this holds for all $f \in \mathcal{F}_2$, we conclude the non-affine bijective trasformation ψ preserves nonlinearity, i.e., $\psi \in \mathcal{P}_2(N)$.

Example 4.3. Let $\psi \in \Theta(2)$ be defined as:

$$\begin{aligned}\psi(x_1, x_2, x_3, x_4) \quad = \quad & (\psi_0(x_1, x_2, x_3, x_4), \psi_1(x_1, x_2, x_3, x_4), \\ & \psi_2(x_1, x_2, x_3, x_4), \psi_3(x_1, x_2, x_3, x_4)),\end{aligned}$$

where,

$$\begin{aligned}\psi_0(x_1, x_2, x_3, x_4) &= 1 \oplus x_1 \oplus x_1x_4 \oplus x_2x_4 \oplus x_3x_4 \oplus x_1x_3x_4 \\ &\quad \oplus x_2x_3x_4 \\ \psi_1(x_1, x_2, x_3, x_4) &= x_2 \oplus x_1x_4 \oplus x_2x_4 \oplus x_3x_4 \oplus x_1x_3x_4 \\ &\quad \oplus x_2x_3x_4 \\ \psi_2(x_1, x_2, x_3, x_4) &= x_1x_2 \oplus x_4 \oplus x_3x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \\ \psi_3(x_1, x_2, x_3, x_4) &= 1 \oplus x_1x_2 \oplus x_3 \oplus x_3x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4.\end{aligned}$$

Then,

$$\begin{aligned}\psi : T_f \quad \longmapsto \quad & (\lambda_0 \oplus AT_f^t \oplus \lambda_{12} f(\alpha_0) f(\alpha_1) \oplus \lambda_{14} f(\alpha_0) f(\alpha_3) \oplus \\ & \lambda_{24} f(\alpha_1) f(\alpha_3) \oplus \lambda_{34} f(\alpha_2) f(\alpha_3) \oplus \\ & \lambda_{134} f(\alpha_0) f(\alpha_2) f(\alpha_3) \oplus \lambda_{234} f(\alpha_1) f(\alpha_2) f(\alpha_3))^t,\end{aligned}$$

where $\lambda_0 = [1001]^t$, $\lambda_{12} = [0011]^t$, $\lambda_{14} = [1100]^t$, $\lambda_{24} = [1100]^t$, $\lambda_{34} = [1111]^t$, $\lambda_{134} = [1111]^t$, $\lambda_{234} = [1111]^t$ and A is the matrix;

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

Again, $\psi \notin \mathcal{A}(2)$. Nevertheless, applying ψ to a function f is rearranging the input arguments of f by an affine bijective transformation and adding some affine functions to the truth table of f which are determined by f itself. Hence, by Theorem 3.1 and Lemma 3.3, ψ does not change the nonlinearity. Since, this is true for all $f \in \mathcal{F}_2$, the non-affine bijective trasformation ψ keeps nonlinearity invariant, i.e., $\psi \in \mathcal{P}_2(N)$.

5. Conclusion

We have presented the exact set of nonlinearity preserving affine bijective transformations by considering the transformations acting on the truth tables of Boolan functions and we showed that there exists a class of nonlinearity preserving non-affine transformations. Furthermore, for $n = 2$, with search results we proved that Proposition 3.8 is in fact necessary and sufficient for preserving nonlinearity, that is the transformations satisfying the conditions in Proposition 3.8 are the only nonlinearity preserving transformations. As a future work, we are planning to pursue this investigation to clearify the whole nonlinearity preserving bijective transformations and examine the contributions of these transformations to construction of Boolean functions with desirible desgin criteria.

References

[1] Shannon C.E., *Communication Theory of Secrecy Systems*, Bell System Technical Journal, Vol. 28: 656-715 (1949).

[2] Hall Jr. M., *Note on the Mathieu group* M_{12}, Arch. Math. 13: 334-340 (1962).

[3] MacWilliams F.J., Sloane N.J.A., *The theory of error-correcting codes*, Amsterdam, New York, Oxford:North-Holland (1978).

[4] Meier W. and Staffelbach O., *Nonlinearity Criteria for cryptographic functions*, Advances in Cryptology - EUROCRYPT'89 (Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, New York 1990) 434: 549-562 (1989).

[5] Preneel B., *Analysis and Design of Cryptographic Hash Functions*, Ph. D. Thesis, COSIC, Katholieke Universiteit Leuven, Belgium, (1993).

[6] Sertkaya İ., *Nonlinearity Preserving Post-Transformations*, M. Sc. Thesis, Institute of Applied Mathematics, Middle East Technical University, Ankara, Turkey, (2004).

[7] Sertkaya İ., Doğanksoy A., *On Nonlinearity Preserving Bijective Transformations*, 2nd National Symposium on Cryptology, Ankara, Turkey, (2006).

CONSTRUCTION OF RESILIENT FUNCTIONS BY THE CONCATENATION OF BOOLEAN FUNCTIONS HAVING NONINTERSECTING WALSH SPECTRA

Selçuk Kavut[1], Melek D. Yücel[1] and Subhamoy Maitra[2]

Abstract. We study the construction of resilient functions satisfying the upper bound on nonlinearity by concatenating Boolean functions such that the positions corresponding to nonzero values in their Walsh spectra are non-overlapping. For this purpose, we suitably modify a steepest descent based search algorithm to obtain 9-variable functions achieving nonlinearity 240 and resiliency order 3. Such functions have been discovered very recently by a particle swarm optimization (PSO) based heuristic search. We independently confirm that it is possible to generate those functions by concatenating properly selected four 7-variable functions with nonlinearity 48 and resiliency order 3, obtained by the steepest descent based search algorithm that we describe in detail. In the process, we show that some resilient functions known by theoretical constructions can also be generated heuristically, even for quite large search spaces as that of $n = 9$. More specifically, we could obtain 9-variable functions having nonlinearity 224 and resiliency order 4 by searching the whole space. Several pairs of such functions with nonintersecting Walsh spectra could be reliably produced, suitable for concatenation to yield 10-variable functions having nonlinearity 480 and resiliency order 4. Moreover, restricting the search space to Rotation Symmetric Boolean Functions (RSBFs), we could demonstrate several 10-variable RSBFs having nonlinearity 488 and correlation immunity of order 2, which have not been demonstrated previously.

[1] Department of Electrical and Electronics Engineering, Middle East Technical University, 06531 Ankara, Türkiye.
email: {kavut, melekdy}@metu.edu.tr
[2] Applied Statistics Unit, Indian Statistical Institute, 203 B T Road, Kolkata 700 108, India.
email: subho@isical.ac.in

J-F. Michon, P. Valarcher, J-B. Yunès (Eds.): BFCA'07

1. Introduction

Boolean functions constitute crucial components playing a central role on the strength of cryptographic systems. Nonlinearity and resiliency (balanced correlation immunity) are two cryptographically significant properties of Boolean functions. The concept of correlation immune Boolean functions was introduced by Siegenthaler in 1984 [27] to withstand a class of divide-and-conquer attacks on certain models of stream ciphers, and these functions are used in stream cipher systems to resist cryptanalytic attacks [28]. On the other hand, a Boolean function is desired to be highly nonlinear, which provides robustness against Best Affine Approximation (BAA) attacks [7] in the case of stream ciphers and linear cryptanalysis [16] in the case of block ciphers.

The construction of cryptographically important Boolean functions has for some time used general purpose heuristic algorithms like simulated annealing, genetic algorithms, tabu search and various forms of hill-climbing. Such attempts were initially made in [18–20]. These attempts provided good but suboptimal results. Subsequently, simulated annealing [13] was used to provide competitive results [3, 11] in terms of nonlinearity and autocorrelation values together for small functions having the number of input values $n \leq 8$. Some of the functions obtained by annealing [5], could be transformed using simple linear change of basis to obtain resilient functions with excellent profiles (i.e., the best possible trade-offs). Supplementing optimization with theory allows the best possible trade-offs between nonlinearity, algebraic degree and correlation immunity for balanced functions on $n \leq 8$ variables. However, in general, for $n \geq 9$, the optimization based algorithms are not competitive since the search space increases super exponentially as n increases. Thus some initial pruning is required before attempting a suitable heuristic search. Very recently, by restricting the search space to the class of Rotation Symmetric Boolean Functions (RSBFs), some 9-variable RSBFs having nonlinearity 241 have been found [10] exploiting the steepest-descent like algorithm [9, 12]. Such highly nonlinear Boolean functions was an important open question in the literature for almost three decades. This kind of search has also provided some other major results in the RSBF class [9, 10]. We use a similar search, which successfully works in a much larger class as investigated here.

The construction of resilient Boolean functions with very good parameters in terms of nonlinearity, autocorrelation, algebraic degree and other cryptographic parameters has received a lot of attention in the literature as evidenced from the papers [1,14,15,21, 23–26,31]. In [24], a tight upper bound on nonlinearity of resilient Boolean functions has been proposed and a list of functions on 7 to 10 variables have been presented in [24, Table 3] which were not known at that time. After that it becomes a challenging question to discover such functions and the papers [10], [21], [14], [22], [30] present some of them. Very recently, the existence of 9-variable, 3-resilient functions having nonlinearity 240 has been demonstrated in [22], which was posed as an open question in Crypto 2000 [24]. These functions could be discovered by a heuristic search based on a modified version of Partial Swarm Optimization (PSO) [29] after an initial pruning of the search space provided by the construction method of the concatenation of Boolean functions having *nonintersecting Walsh spectra* [21, 22, 24, 31]. This method was first noted in Crypto 2000 [24, Lemma 7] in construction of 8-variable, 3-resilient Boolean functions having nonlinearity 112 by concatenating two 7-variable 3-resilient functions each with nonlinearity 48. This has been generalized in [31] presenting n-variable, m-resilient Boolean functions with maximum possible nonlinearity $2^{n-1} - 2^{m+1}$ for $\frac{2n-7}{3} \leq m \leq n-2$, and studied further in [21] providing Boolean functions on $n = 7 + 3i$ variables ($i \geq 0$) with order of resiliency $m = 2 + 2i$. The major result of 9-variable, 3-resilient Boolean functions having nonlinearity 240 is presented in [22], which provides the motivation of this paper. Specifically, given two n-variable Boolean functions g and h having nonlinearities $nl(g)$ and $nl(h)$ respectively, the concatenation $f = (g \parallel h)$ has $nl(f) \geq nl(g) + nl(h)$, but in general the equality occurs since the Walsh spectra of g and h are not disjoint in general. On the other hand, if g and h have nonintersecting Walsh spectra, then $nl(f) = 2^{n-1} + \min(nl(g), nl(h))$, that is, the minimum nonlinearity of g and h determines $nl(f)$. The situation can be extended to the concatenation of more than two functions as in [22]. In [22], however, description of the PSO based heuristic search is not included. Here we suitably modify a steepest-descent based iterative search algorithm appeared first in [12], which independently confirms that such functions can be generated by concatenating

properly selected four 7-variable, 3-resilient functions with nonlinearity 48 having nonintersecting Walsh spectra. From this motivation, a natural question is whether it is possible to construct an 11-variable, 4-resilient Boolean function having nonlinearity 992, which is still an unknown function in the literature, by concatenating sixteen 7-variable, 4 resilient functions with nonlinearity 32. In Section 3.3, we show that such a construction does not exist.

In the process, we could reliably generate 9-variable, 4-resilient functions having nonlinearity 224 by searching the whole space. We could also find several pairs of these functions such that the concatenation of each pair results in a 10-variable, 4-resilient function having nonlinearity 480. In addition to the theoretical constructions [21, 24] satisfying the upper bound on nonlinearity of resilient Boolean functions, our results show that such functions can also be generated by heuristic search.

The construction of 10-variable, 2-resilient Boolean functions having nonlinearity 488 is an open problem since Crypto 2000 [24]. Here we present an important result which provides further vision in this direction. Restricting the search space to the class of Rotation Symmetric Boolean Functions (RSBFs) and exploiting a suitable cost function, we could demonstrate several 10-variable RSBFs having nonlinearity 488 and correlation immunity of order 2, which have not been demonstrated previously. Then, the problem becomes whether it is possible to transform such RSBFs to the 2-resilient functions using linear change of basis. On the other hand, considering the efficient enumeration of some 9-variable RSBFs possessing some important cryptographic properties [6, 17] and 10-variable rotation symmetric bent functions [8], we motivate the researchers in enumeration of 10-variable RSBFs having nonlinearity 488 and correlation immunity of order 2 (or resiliency order 2, if exist) using a reasonable amount of computational power.

In the following section the basic definitions related to Boolean functions are provided. In Section 3, we present our search strategy in detail and provide our method used to obtain correlation immune RSBFs. Section 4 is devoted to the conclusions.

2. Preliminaries on Boolean Functions

Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function that produces a single-bit result for each possible combination of values from n Boolean variables. The *truth table* of a Boolean function $f(x_1, \ldots, x_n)$ is a binary string of length 2^n, $f = [f(0,0,\cdots,0), f(1,0,\cdots,0), f(0,1,\cdots,0), \ldots, f(1,1,\cdots,1)]$. The *Hamming weight* of a Boolean function f is the number of 1's in its truth table and denoted by $wt(f)$. An n-variable Boolean function f is said to be *balanced* if its truth table contains an equal number of 0's and 1's, i.e., $wt(f) = 2^{n-1}$. Also, the *Hamming distance* between equidimensional Boolean functions f and g is defined by $d(f,g) = wt(f \oplus g)$.

Algebraic Normal Form and Degree. An n-variable Boolean function $f(x_1, \ldots, x_n)$ can be considered to be a multivariate polynomial over $GF(2)$. This polynomial can be expressed as a sum of products representation of all distinct k-th order products ($0 \leq k \leq n$) of the variables. More precisely, $f(x_1, \ldots, x_n)$ can be written as

$$a_0 \oplus \bigoplus_{1 \leq i \leq n} a_i x_i \oplus \bigoplus_{1 \leq i < j \leq n} a_{ij} x_i x_j \oplus \ldots \oplus a_{12\ldots n} x_1 x_2 \ldots x_n,$$

where the coefficients $a_0, a_{ij}, \ldots, a_{12\ldots n} \in \{0,1\}$. This representation of f is called the *algebraic normal form* (ANF) of f. The number of variables in the highest order product term with nonzero coefficient is called the *algebraic degree*, or simply the degree of f and denoted by $deg(f)$.

Affine and Linear Boolean Functions. A Boolean function $f(\mathbf{x})$ having degree at most one is called an *affine* function of $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$. Its ANF is given by

$$f(\mathbf{x}) = w_1 x_1 \oplus w_2 x_2 \oplus \ldots \oplus w_n x_n \oplus c = \mathbf{w} \cdot \mathbf{x} \oplus c,$$

where $c \in \{0,1\}$, $\mathbf{w} = (w_1, \ldots, w_n) \in \{0,1\}^n$, and $\mathbf{w} \cdot \mathbf{x}$ represents the inner product of $\mathbf{w}$ and $\mathbf{x}$. An affine function with the constant term $c = 0$ is called *linear*. The set of all n-variable affine (respectively linear) functions is denoted by $A(n)$ (respectively $L(n)$).

Walsh Hadamard Transform. For a Boolean function f the Walsh Hadamard transform is a real valued function over $\{0,1\}^n$

which is defined as

$$W_f(\mathbf{w}) = \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\oplus \mathbf{x}\cdot\mathbf{w}}.$$

We refer to the vector $\boldsymbol{W}_f = [W_f(0,0,\cdots,0),\ W_f(1,0,\cdots,0),\ W_f(0,1,\cdots,0),\ \ldots,\ W_f(1,1,\cdots,1)]$ as the *Walsh spectrum* , or simply the spectrum of the function f. The Boolean functions f and g are said to have *nonintersecting Walsh spectra* [24, Lemma 7] if and only if $W_f(\mathbf{w}) \neq 0 \Rightarrow W_g(\mathbf{w}) = 0$ and $W_g(\mathbf{w}) \neq 0 \Rightarrow W_f(\mathbf{w}) = 0$ for all $\mathbf{w} \in \{0,1\}^n$.

Nonlinearity. The nonlinearity of an n-variable Boolean function f is

$$nl(f) = \min_{g\in A(n)} (d(f,g)),$$

i.e, the minimum distance from the set of all n-variable affine functions. In terms of Walsh spectrum, the nonlinearity of f is given by

$$nl(f) = 2^{n-1} - \frac{1}{2} \max_{\mathbf{w}\in\{0,1\}^n} |W_f(\mathbf{w})|.$$

Correlation Immunity and Resiliency. Zhen and Massey [33] have provided a spectral characterization of correlation immune functions, which we use as the definition here. A Boolean function f is m-th order correlation immune (respectively m-resilient) if and only if its Walsh transform satisfies

$$W_f(\mathbf{w}) = 0, \text{ for } 1 \leq wt(\mathbf{w}) \leq m \text{ (respectively } 0 \leq wt(\mathbf{w}) \leq m).$$

Parseval's Theorem. It states that for an n-variable Boolean function f, the sum of squared Walsh spectrum is constant and equal to 2^{2n}:

$$\sum_{\mathbf{w}\in\{0,1\}^n} (W_f(\mathbf{w}))^2 = 2^{2n}.$$

Autocorrelation Function. The autocorrelation function of a Boolean function f is given by

$$r_f(\mathbf{d}) = \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\oplus f(\mathbf{x}\oplus\mathbf{d})},$$

where $\mathbf{d} \in \{0,1\}^n$. The autocorrelation value having maximum magnitude (excluding the value at the origin which is equal to 2^n) is also known as the absolute indicator [32] and denoted as:

$$\Delta_f = \max_{\mathbf{d}\in\{0,1\}^n, \mathbf{d}\neq(0,\ldots,0)} |r_f(\mathbf{d})|.$$

Following the notation used in [24], we define the profile of a Boolean function by (n, m, d, σ) as its (input variable length, resiliency order, degree, nonlinearity).

3. Search Algorithm

3.1. Spectral Inversion Method

In [4], an unusual approach in the design of Boolean functions has been introduced in which Walsh spectra is exploited as the search space instead of truth tables. The motivation in this approach results from the fact that various important cryptographic criteria (balance, nonlinearity, correlation immunity, resiliency) are defined in terms of the Walsh values of that function. The basic idea is to start with a set of Walsh values that satisfy the desired cryptographic properties. In general, it may be rather hard to generate an initial set of Walsh values, but in some cases it is relatively easier through the following Lemma (proved in [24]) and arbitrarily fixing the first bit in the truth table.

Lemma 1: [24] Let $n \geq 3$ and $m > \lfloor \frac{n}{2} \rfloor - 2$. The spectrum of any $(n, m, -, 2^{n-1} - 2^{m+1})$ function is necessarily three valued $(0, \pm 2^{m+2})$.

In addition, it has been known that the algebraic degree of the function $(n, m, d, 2^{n-1} - 2^{m+1})$ is always maximum and equal to $d = n - m - 1$ [2]. Let us now illustrate how to generate a starting set of Walsh values by considering Boolean functions with profile (7, 3, 3, 48). The corresponding Walsh values must be either 0 or ± 32 by Lemma 1. Using Parseval's theorem it is found that the number of nonzero Walsh values (± 32's) is equal to 16 ($= \frac{2^{2\times 7}}{2^{2\times 5}}$). Further, since the following inverse Walsh transform relation

$$2^n \times (-1)^{f(0,\ldots,0)} = \sum_{\mathbf{w}\in\{0,1\}^n} W_f(\mathbf{w})$$

defines the value of $f(0,0,\ldots,0)$, arbitrarily fixing this to be zero constrains the distribution of the nonzero Walsh values to: 10 many '+32's and 6 many '−32's. Then, zero Walsh values are placed corresponding to $\mathbf{w}$'s with weights satisfying $0 \leq wt(\mathbf{w}) \leq 3$ and the remaining Walsh values (i.e., ten '+32's, six '−32's and forty eight '0's) are arbitrarily allocated to the remaining positions. This initial set of Walsh values is simply a permutation of the spectrum of a (7, 3, 3, 48) function. However, since such a permuted set of Walsh values is not guaranteed to be the Walsh spectrum for some Boolean function, the problem reduces to finding a suitable permutation such that when it is applied to this set, the resulting function obtained by applying the inverse Walsh transform to the permuted spectrum is Boolean. While a few permutations will correspond to Boolean functions, most will not. With each permutation we associate a cost that indicates how far the permuted spectrum is from the spectrum of a valid Boolean function. The objective cost function, to be minimized, is based on Titsworth's theorem [7], which states that $\boldsymbol{W}_f$ is the Walsh spectrum of a Boolean function if and only if

$$\sum_{\mathbf{w}\in\{0,1\}^n} W_f(\mathbf{w})W_f(\mathbf{s}\oplus\mathbf{w}) = 2^{2n}\delta(\mathbf{s}).$$

where $\delta(\mathbf{s}) = 1$ if $s = (0,\ldots,0) \in \{0,1\}^n$ and $\delta(\mathbf{s}) = 0$ otherwise. The theorem suggests the following cost function as used in [4,22]:

$$Cost(\boldsymbol{W}_f) = \sum_{\mathbf{s}\neq(0,\ldots,0)} (|\sum_{\mathbf{w}} W_f(\mathbf{w})W_f(\mathbf{s}\oplus\mathbf{w})|)^R.$$

For $\mathbf{s} = (0,\ldots,0)$, since the inner sum is constant and equal to 2^{2n} for all permutations, this term is not considered. The cost function punishes deviations from zero and is equal to zero for a Boolean function. We set $R = 1$ in our experiments.

3.2. **The Steepest-Descent Like Search Strategy**

The search strategy that we extend for the construction of resilient functions by the spectral inversion method uses a steepest-descent based iterative algorithm [12], where each iteration step has the input set of Walsh values $W_f(\mathbf{w})$ (in short, the vector $\boldsymbol{W}$) and the output set of Walsh values $\boldsymbol{W}_{min}$. At each iteration step, the cost function is calculated within a pre-defined neighborhood

of $\boldsymbol{W}$ and the set of Walsh values having the smallest cost is chosen as the iteration output $\boldsymbol{W}_{min}$. In some rare cases, the cost $\boldsymbol{W}_{min}$ may be larger than or equal to the cost of $\boldsymbol{W}$. This is the crucial part of the search strategy, which provides the ability to escape from local minima and its distinction from the steepest-descent like algorithm.

In order to carry out the search efficiently, a proper definition of the neighborhood is essential. A large number of neighbors can make each iteration very slow due to the computational power required to calculate corresponding cost values, whereas a small set of neighbors can cause an incomplete search in each iteration missing possible candidates having less cost. Imagining such a trade-off, we define the neighborhood as follows. Let us consider again the Walsh spectrum of a (7, 3, 3, 48) function. As stated before, a candidate spectrum, $\boldsymbol{W}$, is formed by placing zero values corresponding to $0 \leq wt(\mathbf{w}) \leq 3$ (these elements remain fixed throughout the search) and arbitrarily allocating 10 values of '+32's, 6 values of '−32's and 48 many '0's to the remaining positions. Denoting the set of positions having zero Walsh values (out of the fixed elements) by S_{zero}, and the set of positions having nonzero Walsh values by $S_{nonzero}$, a neighbor of $\boldsymbol{W}$ is obtained simply replacing an element in S_{zero} by an element in $S_{nonzero}$. Hence, in the neighborhood of $\boldsymbol{W}$, there are $|S_{zero}| \times |S_{nonzero}|$ $(= 48 \times 16 = 768)$ many different permuted sets of $\boldsymbol{W}$.

The search strategy given below starts with an initial set of Walsh values, $\boldsymbol{W}_{initial}$, which is an arbitrary permutation of the desired Walsh spectrum, and stops after a fixed number of iterations, N. At each iteration, $|S_{zero}| \times |S_{nonzero}|$ distinct Walsh spectra within the predefined neighborhood, each of which is shown by $\boldsymbol{W}_{swapped}$, are visited by storing the cost value $cost_{swapped}$ in $COST$, and the corresponding set of Walsh values itself in SET_W. Among the stored cost values, the minimum one, $cost_{min}$, is chosen, and the respective set of Walsh values, $\boldsymbol{W}_{min}$, is obtained from SET_W as the candidate of the step output. If the candidate $\boldsymbol{W}_{min}$ is already in $STORE$, which contains all previous iteration outputs, then this candidate $\boldsymbol{W}_{min}$ and its cost are removed from SET_W and $COST$ respectively. The minimum cost value is searched again in $COST$ among the remaining cost values to find the respective new candidate for $\boldsymbol{W}_{min}$.

Algorithm 1

$\boldsymbol{W} = \boldsymbol{W}_{initial}$;
for(int $k = 0$; $k < N$; $k++$){
    for(int $i = 0$; $i < |S_{zero}|$; $i++$){
        for(int $j = 0$; $j < |S_{nonzero}|$; $j++$){
            Swap i-th element in S_{zero} and j-th element in $S_{nonzero}$
            $SET_W[\ i\ ][\ j\ ] = \boldsymbol{W}_{swapped}$
            $COST[\ i\ ][\ j\ ] = cost_{swapped}$
        }
    }
    Find $cost_{min}$ (minimum $cost_{swapped}$ in $COST$), and $\boldsymbol{W}_{min}$ (respective $\boldsymbol{W}_{swapped}$ in SET_W)
    while($\boldsymbol{W}_{min}$ is already in $STORE$){
        Remove $cost_{min}$ from $COST$, and $\boldsymbol{W}_{min}$ from SET_W
        Find $cost_{min}$ in $COST$, and $\boldsymbol{W}_{min}$ in SET_W
    }
    $STORE[k] = \boldsymbol{W}_{min}$
    $\boldsymbol{W} = \boldsymbol{W}_{min}$
}

We can store the iteration outputs in $STORE$ efficiently to save the memory requirements as follows. Consider the above example of (7, 3, 3, 48) functions. Since the 64 elements are fixed by placing zero values corresponding to $0 \leq wt(\mathbf{w}) \leq 3$, it is enough to store the remaining 64 positions and corresponding Walsh values. However, the positions corresponding to the values of '+32's can be stored in a 64-bit vector by giving one if the value is 32 and giving zero otherwise. Similarly, the positions corresponding to the values of '-32's can be stored in another 64-bit vector. Therefore, in the case of (7, 3, 3, 48) functions, it is possible to store an iteration output by four 32-bit words using a 32-bit microprocessor or two 64-bit words using a 64-bit microprocessor. Similarly, in the case of (9, 4, 4, 224) functions, an iteration output can be stored in a sixteen 32-bit words or in an eight 64-bit words.

In Algorithm 1, the value of $cost_{swapped}$ can be calculated in an efficient manner as follows. The cost value of an iteration input set of Walsh values $\boldsymbol{W}$ can be stored in an array $\boldsymbol{D}$ such that each

array element corresponds to a difference vector $\mathbf{s}$ as given below:

$$Cost(\boldsymbol{W}) = \sum_{\mathbf{s}\neq(0,\ldots,0)} |D(\mathbf{s})|,$$

where $D(\mathbf{s}) = \sum_{\mathbf{w}} W(\mathbf{w})W(\mathbf{s}\oplus\mathbf{w})$. Arbitrarily representing a zero Walsh value at position $\mathbf{u}$ as $W(\mathbf{u})$, and a nonzero Walsh value at position $\mathbf{v}$ as $W(\mathbf{u})$, $D(\mathbf{s})$ can be expressed separating the terms having either $W(\mathbf{v})$ or $W(\mathbf{u})$ as in the following form:

$$\begin{aligned} D(\mathbf{s}) &= \textstyle\sum_{\mathbf{w}\neq\mathbf{v},\mathbf{w}\neq\mathbf{v}\oplus\mathbf{s},\mathbf{w}\neq\mathbf{u},\mathbf{w}\neq\mathbf{u}\oplus\mathbf{s}} W(\mathbf{w})W(\mathbf{s}\oplus\mathbf{w}) \\ &\quad + 2(W(\mathbf{v})W(\mathbf{s}\oplus\mathbf{v})) + 2(W(\mathbf{u})W(\mathbf{s}\oplus\mathbf{u})), \\ &= \textstyle\sum_{\mathbf{w}\neq\mathbf{v},\mathbf{w}\neq\mathbf{v}\oplus\mathbf{s},\mathbf{w}\neq\mathbf{u},\mathbf{w}\neq\mathbf{u}\oplus\mathbf{s}} W(\mathbf{w})W(\mathbf{s}\oplus\mathbf{w}) \\ &\quad + 2(W(\mathbf{v})W(\mathbf{s}\oplus\mathbf{v})). \end{aligned}$$

The simplification in the expression is due to the fact that $W(\mathbf{u}) = 0$. Notice that for the difference vector $\mathbf{s} = \mathbf{u}\oplus\mathbf{v}$, the value of $D(\mathbf{u}\oplus\mathbf{v})$ remains the same after swapping $W(\mathbf{u})$ and $W(\mathbf{v})$. In the following, we give an efficient method to calculate the values of array elements changing for the remaining $\mathbf{s}$ vectors, after the replacement of $W(\mathbf{u})$ and $W(\mathbf{v})$. Let us denote the array corresponding to the set of Walsh values $\boldsymbol{W}_{swapped}$, obtained by swapping $W(\mathbf{u})$ and $W(\mathbf{v})$, as $\boldsymbol{D}_{swapped}$. Then, the following relation between $D(\mathbf{s})$ and $D_{swapped}(\mathbf{s})$ is found (where the subscript *swapped* is denoted by *sw*):

$$\begin{aligned} D_{sw}(\mathbf{s}) &= \textstyle\sum_{\mathbf{w}\neq\mathbf{v},\mathbf{w}\neq\mathbf{v}\oplus\mathbf{s},\mathbf{w}\neq\mathbf{u},\mathbf{w}\neq\mathbf{u}\oplus\mathbf{s}} W_{sw}(\mathbf{w})W_{sw}(\mathbf{s}\oplus\mathbf{w}) \\ &\quad +2(W_{sw}(\mathbf{v})W_{sw}(\mathbf{s}\oplus\mathbf{v})) + 2(W_{sw}(\mathbf{u})W_{sw}(\mathbf{s}\oplus\mathbf{u})), \\ &= \textstyle\sum_{\mathbf{w}\neq\mathbf{v},\mathbf{w}\neq\mathbf{v}\oplus\mathbf{s},\mathbf{w}\neq\mathbf{u},\mathbf{w}\neq\mathbf{u}\oplus\mathbf{s}} W(\mathbf{w})W(\mathbf{s}\oplus\mathbf{w}) \\ &\quad + 2(W(\mathbf{u})W(\mathbf{s}\oplus\mathbf{v})) \ +2(W(\mathbf{v})W(\mathbf{s}\oplus\mathbf{u})), \\ &= \textstyle\sum_{\mathbf{w}\neq\mathbf{v},\mathbf{w}\neq\mathbf{v}\oplus\mathbf{s},\mathbf{w}\neq\mathbf{u},\mathbf{w}\neq\mathbf{u}\oplus\mathbf{s}} W(\mathbf{w})W(\mathbf{s}\oplus\mathbf{w}) \\ &\quad + 2(W(\mathbf{v})W(\mathbf{s}\oplus\mathbf{u})), \end{aligned}$$

$$= D(\mathbf{s}) - 2(W(\mathbf{v})W(\mathbf{s} \oplus \mathbf{v}) - W(\mathbf{v})W(\mathbf{s} \oplus \mathbf{u})).$$

Consequently, $Cost(\boldsymbol{W}_{swapped})$ $(= cost_{swapped})$ is calculated using the absolute values of $D_{swapped}(\mathbf{s})$ (except $D_{swapped}(0, \ldots, 0)$) for which $D_{swapped}(\mathbf{u} \oplus \mathbf{v}) = D(\mathbf{u} \oplus \mathbf{v})$. Hence, considering the number of only multiplications used, as it is the most time consuming operation, the complexity for the calculation of the cost reduces from $O(2^{2n})$ to $O(3 \cdot 2^n)$, which provides a remarkable efficiency.

For (7, 3, 3, 48) functions, we have carried out 100 runs each with $N = 100$ iterations, which generates 90 successes in a few seconds using a computer system with Pentium IV 2.8 GHz processor and 256 MB RAM. This shows the ease of generation of such functions by the steepest-descent like search strategy. For (9, 4, 4, 224) functions, we have carried out 150 runs setting $N = 4000$, which results in 6 successes (due to the super exponential increase of the search space) among the generated 49,152,000 $(= |S_{zero}| \times |S_{nonzero}| \times N = 192 \times 64 \times 4000)$ permuted sets of Walsh values. With the same computer system, a typical run of Algorithm 1 takes 6 minutes and 50 seconds for $n = 9$. Some of the attained 9-variable functions are given in Appendix A.

3.3. **Search for the Nonintersecting Walsh Spectra**

The motivation in [22] to attain a Boolean function having profile (9, 3, 5, 240) results from the observation of the following Lemma, which follows directly from the basic definition of the Walsh transform.

Lemma 2: Let f be an $(n+2)$-variable Boolean function obtained from the concatenation of n-variable Boolean functions f_1, f_2, f_3 and f_4, i.e., $f = [f_1 \parallel f_2 \parallel f_3 \parallel f_4]$. Then the Walsh spectrum $\boldsymbol{W}_f$, of f is given by

$$\begin{aligned} = [&\boldsymbol{W}_{f_1} + \boldsymbol{W}_{f_2} + \boldsymbol{W}_{f_3} + \boldsymbol{W}_{f_4} \parallel \boldsymbol{W}_{f_1} - \boldsymbol{W}_{f_2} + \boldsymbol{W}_{f_3} - \boldsymbol{W}_{f_4} \\ &\parallel \boldsymbol{W}_{f_1} + \boldsymbol{W}_{f_2} - \boldsymbol{W}_{f_3} - \boldsymbol{W}_{f_4} \parallel \boldsymbol{W}_{f_1} - \boldsymbol{W}_{f_2} - \boldsymbol{W}_{f_3} + \boldsymbol{W}_{f_4}]. \end{aligned}$$

Keeping in mind that the degree of the function $(n, m, d, 2^{n-1} - 2^{m+1})$ is always maximum and equal to $d = n - m - 1$ [2], Lemmas 1 and 2 indicate that it is possible to construct an $(n, m, n - m - 1, 2^{n-1} - 2^{m+1})$ function where $n \geq 3$ and $m > \lfloor n/2 \rfloor - 2$ from the

concatenation of four $(n-2, m, n-m-3, 2^{n-3}-2^{m+1})$ functions with nonintersecting Walsh spectra, if such functions exist.

Consequently, the search for (9, 3, 5, 240) function reduces to find four (7, 3, 3, 48) functions with nonintersecting Walsh spectra, which helps in pruning the search space dramatically compared to the direct search for a (9, 3, 5, 240) function. We start searching for the first (7, 3, 3, 48) function f_1 using Algorithm 1, then proceed (as in [22]) by finding the next (7, 3, 3, 48) function $f_i, i = 2, 3, 4$ with the following additional conditions on its Walsh spectrum:

$$W_{f_{i-j}}(\mathbf{w}) \neq 0 \Rightarrow W_{f_i}(\mathbf{w}) = 0 \text{ and } W_{f_i}(\mathbf{w}) \neq 0 \Rightarrow W_{f_{i-j}}(\mathbf{w}) = 0,$$

for all $\mathbf{w} \in \{0,1\}^n$ and $1 \leq j \leq i-1$. For the search of f_2 and f_3, we restart Algorithm 1 with a slight modification. Specifically, in the search for f_2 (respectively f_3), the positions of the nonzero Walsh values of $\boldsymbol{W}_{f_1}$ (respectively $\boldsymbol{W}_{f_1}$ and $\boldsymbol{W}_{f_2}$) remain fixed in addition to the positions corresponding to $0 \leq wt(\mathbf{w}) \leq 3$, and hence the set S_{zero} contains the remaining 32 (respectively 16) elements. However, notice that the set S_{zero} is empty for the case of f_4, since all the elements except those in $S_{nonzero}$ (which contains ten '+32's, six '−32's) are fixed. Therefore, the search for f_4 is carried out by considering possible permutations of the elements in $S_{nonzero}$. The number of all these permutations is reasonably small (8008 permutations), and hence an exhaustive search is possible for f_4.

Using such an approach in our experiments, we have carried out 400 trials setting $N = 100$ for each successive search of the functions f_1, f_2 and f_3, which generates 7 successes in one minute. Although there is no efficiency analysis in [22], for comparison purposes, we note that in [22] it is reported that two examples of such functions are generated in a few minutes using the PSO based search algorithm, which confirms the efficiency of our search strategy. We observe that generally the functions f_2 and f_3 are attained in the first or second runs of the search algorithm during a success. Some of the resulted (9, 3, 5, 240) functions are given in Appendix A.

In a similar way, we could construct several (10, 4, 5, 480) functions by the concatenation of two (9, 4, 4, 224) functions f_1 and f_2 having nonintersecting Walsh spectra. More specifically, f_1 is obtained using Algorithm 1 directly, and subsequently f_2 is attained

TABLE 1. The number of runs for f_1 and f_2 yielding (10, 4, 5, 480) functions.

	number of runs		
search for f_1	8	25	30
search for f_2	40	39	12

adding the constraint such that the Walsh values of $\boldsymbol{W}_{f_2}$ are zero in places where the Walsh values of $\boldsymbol{W}_{f_1}$ are nonzero, which requires a slight modification of Algorithm 1 as before. Some successive number of runs resulting such pairs of (9, 4, 4, 224) functions are given in Table 1. For instance, from the second column, it is seen that f_1 is found in the 8-th run of Algorithm 1, and then f_2 is obtained in the 40-th run using Algorithm 1 which is properly modified by fixing the positions corresponding to the nonzero Walsh values of $\boldsymbol{W}_{f_1}$.

We have implemented 100 trials with the number of iterations $N = 4000$, which results in 3 successes producing (10, 4, 5, 480) functions. Some of these functions are provided in Appendix A.

Considering such applications of the nonintersecting spectra in the construction of resilient functions, one can naturally ask whether it is possible to construct a (11, 4, 6, 992) function, which is still an unknown function in the literature, by concatenating sixteen (7, 4, 2, 32) functions. The following proposition shows that such a construction does not exist.

Proposition 1. Construction of the function (11, 4, 6, 992) obtained by the concatenation of sixteen (7, 4, 2, 32) functions having either nonintersecting or intersecting Walsh spectra does not exist.

Proof. A (7, 4, 2, 32) function has only 29 positions for a nonzero Walsh value to be placed corresponding to $wt(\mathbf{w}) > 4$. Lemma 2 implies that the concatenation of sixteen (7, 4, 2, 32) functions provides at most $29 \times 16 = 464$ nonzero values in the resulted Walsh spectrum. This contradicts that the (11, 4, 6, 992) function has $2^{10} = 1024$ nonzero values each having ± 64 in the Walsh spectrum due to the Lemma 1 and Parseval's theorem.

3.4. **Search for (10, 2, -, 488) RSBFs**

The set of Rotational Symmetric Boolean Functions (RSBFs) is interesting to look into as the space is much smaller ($\approx 2^{\frac{2^n}{n}}$) than the total space of Boolean functions (2^{2^n}) and the set contains functions with very good cryptographic properties. We refer the interested reader to [10], and the references therein. We note only that in RSBFs, all indices which are rotationally equivalent have identical values of f. We use the same search algorithm as in [10] within the space of RSBFs, however with a different cost function which is more suitable for highly nonlinear (balanced) correlation immune functions than the one used in [10]. The result of such an effort produced results which had not been demonstrated previously.

In [5], simulated annealing based search algorithm was used to derive balanced correlation immune functions with high nonlinearity exploiting the following cost function:

$$Cost(f) = \sum_{wt(\mathbf{w})\leq m} |W_f(\mathbf{w})|^R + A \times \max_w |W_f(\mathbf{w})|.$$

Here only the nonzero Walsh values corresponding to $wt(\mathbf{w}) \leq m$ are punished to satisfy m-th order resiliency, while A is a weighting constant for the nonlinearity component. This enables (balanced) correlation immunity and nonlinearity to be optimized simultaneously by the search algorithm. In our experiments, the search strategy in [10] is applied in the space of 10-variable RSBFs setting $m = 2$, $R = 3$, and $A = 100$, which prove successful. Note that the search space in [10] is expressed in terms of truth tables, and hence, we do not exploit the spectral inversion method in this case. As for the algorithmic information, with the same computer system used before, a typical run takes 36 seconds for which the iteration number is 40,000. In 100 runs, 122 many 10-variable RSBFs having nonlinearity 488, and correlation immunity of order 2 are produced. This is the first time showing the existence of such important functions. Some of these functions are presented in Appendix A. We couldn't obtain balanced version of these functions, which was posed as an open question in Crypto 2000 [24].

4. Conclusions

We investigate the construction of resilient functions by concatenating Boolean functions with nonintersecting Walsh spectra, obtained by a steepest-descent based search algorithm that we have suitably modified for the problem. We have attained (9, 3, 5, 240) functions, which have been found very recently by an independent heuristic search [22], based on a modified version of particle swarm optimization (PSO). We independently confirm that it is possible to generate these functions by concatenating properly selected four (7, 3, 3, 48) functions having nonintersecting Walsh spectra. We provide the details of the modified steepest-descent like search algorithm, which was absent for the heuristic search [22] based on PSO. In the process, we could reliably obtain (9, 4, 4, 224) functions by searching the whole space; then, we could find several pairs of such functions yielding (10, 4, 5, 480). These functions provide further vision in addition to some known theoretical constructions. Furthermore, we demonstrate several 10-variable RSBFs having nonlinearity 488 and correlation immunity of order 2, which have not been demonstrated previously.

References

[1] P. Camion, C. Carlet, P. Charpin, and N. Sendrier. On correlation immune functions. In *Advances in Cryptology – CRYPTO'91*, LNCS, No. 576, Springer Verlag, pp. 86–100, 1992.

[2] C. Carlet. On the coset of weight divisibility and nonlinearity of resilient and correlation immune functions. In *Advances in Cryptography CRYPTO 1991*, pp. 86–100, Springer Verlag, 1992.

[3] J. A. Clark and J. L. Jacob. Two-stage optimization in the design of Boolean functions. In *ACISP 2000*, number 1841 in Lecture Notes in Computer Science, pages 242–254. Springer-Verlag, 2000.

[4] J. Clark, J. Jacob, S. Maitra and P. Stănică. Almost Boolean functions: The design of Boolean functions by spectral inversion. *Computational Intelligence*, pages 450–462, Volume 20, Number 3, 2004.

[5] J. Clark, J. Jacob, S. Stepney, S. Maitra and W. Millan. Evolving Boolean functions satisfying multiple criteria. In *INDOCRYPT 2002*, Volume 2551 in Lecture Notes in Computer Science, pages 246–259, Springer Verlag, 2002.

[6] D. K. Dalai, S. Maitra and S. Sarkar. Results on Rotation Symmetric Bent Functions. In *Second Workshop on Boolean Functions: Cryptography and Applications, BFCA 06*, LIFAR, University of Rouen, France, March 13–15, 2006.

[7] C. Ding, G. Xiao, and W. Shan. *The Stability Theory of Stream Ciphers*. Number 561 in Lecture Notes in Computer Science. Springer Verlag, 1991.

[8] S. Kavut, S. Maitra, S. Sarkar, and M. D. Yücel. Enumeration of 9-variable Rotation Symmetric Boolean Functions having nonlinearity $>$ 240. In *Progress in Cryptology – Indocrypt 2006*, LNCS 4329, pages 266–279, Springer Verlag, December 2006.

[9] S. Kavut, S. Maitra and M. D. Yücel. Autocorrelation spectra of balanced Boolean functions on an odd number input variables with maximum absolute value $< 2^{\frac{n+1}{2}}$. In *Second International Workshop on Boolean Functions: Cryptography and Applications, BFCA 06*, March 13–15, 2006, LIFAR, University of Rouen, France.

[10] S. Kavut, S. Maitra and M. D. Yücel. There exist Boolean functions on n (odd) variables having nonlinearity $> 2^{n-1} - 2^{\frac{n-1}{2}}$ if and only if $n > 7$. IACR eprint server, http://eprint.iacr.org/2006/181, 28 May, 2006 (accepted for publication in the IEEE Transactions on Information Theory with the title "Search for Boolean Functions with Excellent Profiles in the Rotation Symmetric Class").

[11] S. Kavut and M. D. Yücel. Improved cost function in the design of Boolean functions satisfying multiple criteria. In *Indocrypt 2003*, pages 121–134, Lecture Notes in Computer Science, Volume 2904, Springer Verlag, 2003.

[12] S. Kavut and M. D. Yücel. A new algorithm for the design of strong Boolean functions. In *First National Cryptology Symposium*, pages 95–105, METU, Ankara, Türkiye, November 18-20, 2005.

[13] S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[14] S. Maitra and E. Pasalic. Further constructions of resilient Boolean functions with very high nonlinearity. *IEEE Transactions on Information Theory*, 48(7):1825–1834, July 2002.

[15] S. Maitra and P. Sarkar. Hamming weights of correlation immune Boolean functions. *Information Processing Letters*, Vol. 71, pp. 149–153, 1999.

[16] M. Matsui Linear cryptanalysis method for DES cipher. In *Advances in Cryptology – EUROCRYPT'93*, Lecture Notes in Computer Science, pages 386–397. Springer Verlag, 1994.

[17] A. Maximov, M. Hell and S. Maitra. Plateaued rotation symmetric Boolean functions on odd number of variables. In *First Workshop on Boolean Functions: Cryptography and Applications, BFCA 05*, University of Rouen, France, March 7–9, 2005.

[18] W. Millan, A. Clark and E. Dawson. An effective genetic algorithm for finding highly nonlinear Boolean functions. In *First International Conference on Information and Communications Security*, number 1334 in Lecture Notes in Computer Science, pages 149–158. Springer Verlag, 1997.

[19] W. Millan, A. Clark and E. Dawson. Heuristic design of cryptographically strong balanced Boolean functions. In *Advances in Cryptology EUROCRYPT'98*, pages 489–499. Springer Verlag LNCS 1403, 1998.

[20] W. Millan, A. Clark and E. Dawson. Boolean function design using hill climbing methods. In *4th Australasian Conference on Information, Security and Privacy*, number 1587 in Lecture Notes in Computer Science, pages 1–11. Springer Verlag, April 1999.

[21] E. Pasalic, S. Maitra, T. Johansson and P. Sarkar. New constructions of resilient and correlation immune Boolean functions achieving upper bound on nonlinearity. In *Workshop on Coding and Cryptography - WCC 2001*, Paris, January 8–12, 2001. Electronic Notes in Discrete Mathematics, Volume 6, Elsevier Science, 2001.

[22] Z. Saber, M. F. Udin and A. Youssef. On the existence of (9, 3, 5, 240) resilient functions. *IEEE Transactions on Information Theory*, 52(5):2269-2270, May 2006.

[23] P. Sarkar and S. Maitra. Construction of nonlinear Boolean functions with important cryptographic properties. In *Advances in Cryptology - EUROCRYPT 2000*, pp. 485–506, 2000.

[24] P. Sarkar and S. Maitra. Nonlinearity bounds and constuction of resilient Boolean functions. In *Advances in Cryptology - Crypto 2000*, pages 515–532, Berlin, 2000. Springer Verlag. Lecture Notes in Computer Science Volume 1880.

[25] P. Sarkar and S. Maitra. Construction of nonlinear resilient Boolean functions using "small" affine functions. In *IEEE Transactions on Information Theory*, Vol. 50, No. 9, pp. 2185–2193, September 2004.

[26] J. Seberry, X. M. Zhang, and Y. Zheng. On constructions and nonlinearity of correlation immune Boolean functions. In *Advances in Cryptology - EUROCRYPT'93*, pp. 181–199, 1994.

[27] T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, IT-30(5):776-780, September 1984.

[28] T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, C-34(1):81-85, January 1985.

[29] Special Issue on Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, Volume 8, number 3, June 2004.

[30] P. Stănică, S. Maitra and J. Clark. Results on rotation symmetric bent and correlation immune Boolean functions. *Fast Software Encryption Workshop (FSE 2004)*, New Delhi, INDIA, LNCS 3017, Springer Verlag, pages 161–177, 2004.

[31] Y. V. Tarannikov. On resilient Boolean functions with maximum possible nonlinearity. In *Progress in Cryptology - INDOCRYPT 2000*, pp. 19-30, 2000.

[32] X. M. Zhang and Y. Zheng. GAC - the criterion for global avalanche characteristics of cryptographic functions. *Journal of Universal Computer Science*, 1(5):316–333, 1995.

[33] X. Guo-Zhen and J. Massey. A spectral characterization of correlation immune combining functions. *IEEE Transactions on Information Theory*, 34(3): 569-571, May 1988.

Appendix A

We present the truth tables in hexadecimal format. The following truth tables are (9, 3, 5, 240) Boolean functions.

```
4B36D26C9CE13987E19C2D93364BC678
625DAD38CBA1516EB58A46D31C76BA85
6996E41B69961BE496961BE46969E41B
3C66C399A5C35A3C9A5665A9959A6A65

3CC35A96996966A5A569C33C665A9996
4BE49636B41B96C9E11B699C1EE46963
3D4AC2B5A2D55D2AC1B63E495E29A1D6
17E4D28BAC3966599A669C35D14B27E8
```

The following truth tables are (9, 4, 4, 224) Boolean functions.

```
346B4BD4A7C869369669BC83C935529E
DA85A53A16D3D82D6996437C3C6AA7C1
C738B887945B69965AC625793AA5C768
2C7953C6E32C1EE193A5EC1AC55A3897

49E396693C3CD629DA1987B463C638C7
B6469669C399837C25BC784B9C636D92
3C96E349E34609F96996346E85B9DE12
69991CB61CE3F606C366CB917A1C21ED
```

The following truth tables are (10, 4, 5, 480) Boolean functions.

```
79B8861B2A56D5C64B46B4E5C5B93A29
5847A7E46DA99239A6B9591AB1464ED6
84ADD752B74528BA79362AC9C29A5D65
7B83287C48F2D70D86D4D52B3D1EA2E1
4AB597C299663C69E51AA15E693CC396
6669969979896867969 63C6983D65EA1
B54A683D6699C3961AE55EA169C33C96
99966966867697989669C3697C29A15E

63C2966D9A5B65A435BCCA43CC1939B6
6C3D9992C3643C9BC92936D633E6C649
9C9D693235A4CA5B6A4395BC66E3934C
936266CD6C9B936496D66929991C6CB3
```

2AD96D92C536936C6D92615E5AA5B887
718D9966E61A86795AA5AC6329D65798
D62558A73AC99669A15EA798956A4E71
1EE2A95685797689C639539CB94668A7

The following truth tables are 10-variable RSBFs having nonlinearity 488 and correlation immunity of order of 2:

E891D712B67B124DCE396BCB520835A3
A5B80E926CDAF58E731D11D50B26D81E
8C63CA8444ED931969F0A6D9EE26D0BD
7E4A56F30353B662459E1829E2C557AC
85F46D4BA599C0713424ACF2D75F12C6
2982AB559C2DA2C7E8ED1D68B6418BF2
3EB9358D622CFF4A101F630B8B78691C
657382F847911C96BD0CB566727ACCA1

FAC9B4869F64947982AB2860C2253F93
C51899DE4C912C51F14C1D624BEAC31F
A5760294C3C6B6A971B497560CB43743
AB0375A152F62D1821CFADDCB40A02EE
C9627A6D01088375B45EA52DDA789897
3E069A61D67E267911F1CE341B3E600A
98CB101F6E33CC476249EE2D5DA65680
4947A5AB89A3F2E09B3111C85558FCE8

ON DIHEDRAL GROUP INVARIANT BOOLEAN FUNCTIONS (EXTENDED ABSTRACT)

Sumanta Sarkar[1], Subhamoy Maitra[2] and Deepak Kumar Dalai[3]

Abstract. In this paper we consider the Boolean functions which are invariant under the action of Dihedral group. We denote these functions as Dihedral Symmetric Boolean Functions (DSBFs). We study theoretical results in this direction and show how that can be used for efficient search in this class. Most interestingly we note that some of the recently found 9-variable Boolean functions having nonlinearity 241 belong to this class.

Keywords: Boolean Functions, Dihedral Group, Group Action, Nonlinearity, Rotational Symmetry, Walsh transform.

1. Introduction

It has been studied in details that the symmetric Boolean functions are in general not of good cryptographic and combinatorial properties [1,4,6,9,10,15–17,23–25,29,30] and on the other hand, the Rotation Symmetric Boolean functions (RSBFs) can produce excellent results [2,3,5,7,8,12–14,18–21,26–28]. Now we look at a class of Boolean functions which is a subset of the class Rotation Symmetric Boolean functions and a superset of the Symmetric

[1] Applied Statistics Unit, Indian Statistical Institute, 203 B T Road, Kolkata 700 108, INDIA, email: sumanta_r@isical.ac.in
[2] Applied Statistics Unit, Indian Statistical Institute, 203 B T Road, Kolkata 700 108, INDIA, email: subho@isical.ac.in
[3] INRIA, Codes, Domaine de Voluceau-Rocquencourt, BP 105 - 78153, Le Chesnay, France. email: Deepak.Dalai@inria.fr

Boolean functions. This class is invariant under the action of Dihedral group.

Recently, 9-variable Boolean functions having nonlinearity 241 (greater than the bent concatenation bound) have been discovered in the class of RSBFs [13, 14]. Interestingly, we study the class of DSBFs (which are invariant under the action of the Dihedral group on $\{0,1\}^n$) and found some of the 9-variable Boolean functions having nonlinearity 241 in this class.

The organization of the paper is as follows. In Section 2 we describe the Dihedral group D_n. In Section 3, we discuss the action of D_n on the set $\{0,1\}^n$ followed by the discussion on the structure of Walsh Spectra of DSBFs.

2. The Dihedral group

We start with the definition of Dihedral group [11, Page 184].

Definition 2.1. A group G which is generated by two elements $a, b \in G$ such that,

(1) $a^n = b^2 = e$, where e is the identity element and $n \geq 3$,
(2) $ba = a^{-1}b$,

is said to be the Dihedral group of degree n and it is denoted as D_n.

The elements of D_n are $\{e, a, a^2, \ldots, a^{n-1}, b, ab, a^2b, \ldots, a^{n-1}b\}$, i.e., $|D_n| = 2n$.

The geometric realization of the Dihedral group D_n is that it is a group of symmetries ($2n$ many) on the regular n-gon denoted by P_n. Let the vertices of P_n be named as $\{1, 2, \ldots, n\}$. Then all the $2n$ many symmetries can be generated by

(1) rotation of P_n with respect to the line passing vertically through the center of P_n at an angle $\frac{2\pi}{n}$,
(2) reflection of P_n about a line passing through a vertex and the center.

D_n is a permutation group and it is a subgroup of the Symmetric group S_n and it contains the cyclic group C_n as a subgroup. The permutation which is one rotation of the vertices $(1, 2, \ldots, n)$ of P_n is denoted by $\sigma_n^1 = \begin{pmatrix} 1 & 2 & \ldots & n-1 & n \\ 2 & 3 & \ldots & n & 1 \end{pmatrix}$ and the reflection of P_n about the line passing through the vertex 1 and the center is the

permutation denoted by $\tau_n = \begin{pmatrix} 1 & 2 & 3 & \dots & n-1 & n \\ 1 & n & n-1 & \dots & 3 & 2 \end{pmatrix}$. Thus $\{\sigma_n^1, \sigma_n^2, \dots, \sigma_n^{n-1}, \sigma_n^n, \sigma_n^1\tau_n, \sigma_n^2\tau_n, \dots, \sigma_n^{n-1}\tau_n, \sigma_n^n\tau_n\}$ are the elements of D_n. Note that σ_n^n is the identity permutation which is $\begin{pmatrix} 1 & 2 & \dots & n \\ 1 & 2 & \dots & n \end{pmatrix}$. We rename the permutations $\tau_n\sigma_n^i$ as ω_n^i for $1 \leq i \leq n$. Let us take D_4 as an example.

Example 2.2. The elements of D_4 are,

$$\sigma_4^1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}, \sigma_4^2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix},$$
$$\sigma_4^3 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix}, \sigma_4^4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix},$$
$$\omega_4^1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}, \omega_4^2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 4 \end{pmatrix},$$
$$\omega_4^3 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}, \omega_4^4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix}.$$

3. Characterization of the Dihedral group action on the set $\{0,1\}^n$

An n-variable Boolean function is a mapping from $V_n = \{0,1\}^n$ to $\{0,1\}$. We denote the set of all n-variable Boolean functions as $\mathcal{B}_n$ and it is clear that $|\mathcal{B}_n| = 2^{2^n}$. Thus it is not feasible to search exhaustively over $\mathcal{B}_n$ for $n \geq 7$ with the currently available computing facility. There have been a lot of attempts to search Boolean functions with good cryptographic properties in some subclasses of $\mathcal{B}_n$ such as Symmetric Boolean functions and Rotation Symmetric Boolean functions (RSBFs). n-variable Symmetric Boolean functions are such functions which are invariant under the action of the Symmetric group S_n and n-variable RSBFs are invariant under the action of Cyclic group C_n. The size of the subclass of all n-variable Symmetric Boolean functions is 2^{n+1} and that of RSBFs is 2^{g_n}, where $g_n = \frac{1}{n}\sum_{k|n}\phi(k)\,2^{\frac{n}{k}}$, ϕ is the Euler's ϕ function. Note that $g_n \approx \frac{2^n}{n}$. Thus one may get tempted to search for good Boolean functions in these classes in order to take the advantage of the small size (with respect to $\mathcal{B}_n$). So far there have been no significant Boolean functions found in the class of Symmetric Boolean functions. On the other hand, the class of RSBFs has been proved to be very rich in terms of containing Boolean

functions with good cryptographic properties. Recently 9-variable Boolean functions with nonlinearity 241 have been found in the class of RSBFs [13, 14]. This question was open for more than three decades. Moreover, the Patterson-Wiedemann functions [20] on ($n \geq 15$)-variables which have the maximum nonlinearity (exceeding the bent concatenation bound) are also RSBFs. So there is enough motivation to search for cryptographically significant Boolean functions in the class of RSBFs. Unfortunately, searching exhaustively over all the n-variable RSBFs becomes difficult for $n > 9$. So one may attempt to find another class of Boolean functions which has size bigger than that of Symmetric Boolean functions and lesser than that of the RSBFs. In this direction we choose the Dihedral group D_n which is a subgroup of the Symmetric group S_n and supergroup of the Cyclic group C_n. We consider the n-variable Boolean functions which are invariant under the action of the Dihedral group D_n and we denote them as Dihedral Symmetric Boolean functions (DSBFs). Let $S(S_n), S(D_n)$ and $S(C_n)$ respectively denote the set of all Symmetric Boolean functions, RSBFs and DSBFs. Since, $S_n \supseteq D_n \supseteq C_n$, we have $S(S_n) \subseteq S(D_n) \subseteq S(C_n) \subseteq \mathcal{B}_n$ (see also Figure 1).

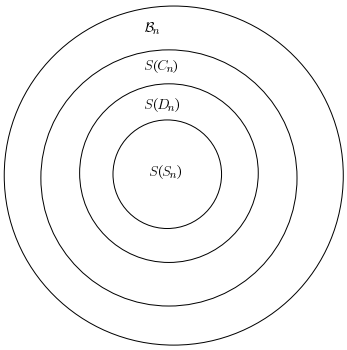


FIGURE 1. The hierarchy of the subclasses of $\mathcal{B}_n$.

We now present a few more definitions.

Definition 3.1 (Group action:). The group action of the group G on the set X is a mapping $\phi : G \times X \to X$ denoted as $g(x)$ or $g \cdot x$, which satisfies the following two axioms.

(1) $(gh) \cdot x = g \cdot (h \cdot x)$, for all $g, h \in G$ and for all $x \in X$.
(2) $e \cdot x = x$, for every $x \in X$, e is the identity element of G.

G is said to act on X and on the other hand X is called a G-set.

Definition 3.2 (**Orbit:**). Let the group G acts on the set X, then the set $G_x = \{g.x | g \in G\}$ is called the orbit generated by x.

These orbits are equivalence classes of X induced by the equivalence relation defined by $x \sim y$ iff there exists a $g \in G$ such that $g \cdot x = y$.

Definition 3.3. Let $X \subseteq V_n$ be a G-set. A Boolean function f is said to be invariant under the action of G if $f(g \cdot x) = f(x)$, for all $g \in G$ and for all $x \in X$.

That means if f is an n-variable Boolean function which is invariant under the action of the group G on the set V_n, then all the outputs corresponding to the inputs of V_n which belong to the same orbit are same. In that case, these functions will form a subclass of $\mathcal{B}_n$ and the size of this class will be 2^m, where m is the number of orbits. Naturally this subclass is smaller in size than $\mathcal{B}_n$. As for example, Symmetric and Rotation Symmetric Boolean functions are invariant respectively under the action of the Symmetric and Cyclic group on the set V_n. The number of orbits can be obtained by the well known Burnside's Lemma.

Lemma 3.4 (Burnside's Lemma)**.** *Let G be a group of permutations acting on the set X. Then the number of orbits induced on X is given by $\frac{1}{|G|}\sum_{g\in G}|fix_X(g)|$, where $fix_X(g) = \{x \in X | g \cdot x = x\}$.*

Let us now discuss how D_n acts on V_n and find the size of the class formed by all n-variable DSBFs using the Burnside's Lemma. The count is known in general, but still we include it here for the special case.

Let $x = (x_1, \ldots, x_n) \in V_n$, then the action of D_n on V_n is as follows.

$$\sigma_n^i(x) = (x_{1+i}, x_{2+i}, \ldots, x_{n+i}), \text{ for } 1 \leq i \leq n,$$
$$\omega_n^i(x) = (x_{1+i}, x_{n+i}, \ldots, x_{2+i}), \text{ for } 1 \leq i \leq n,$$

where the value $k + i$ takes the value $k + i \bmod n$ for $1 \leq i, k \leq n$ with the only exception that it takes the value n if it is a multiple of n.

Theorem 3.5. *Let d_n be the total number of orbits induced by the Dihedral group D_n acting on V_n. Then*

$$d_n = \frac{g_n}{2} + l,$$

where, $g_n = \frac{1}{n}\sum_{t|n}\phi(t)2^{\frac{n}{t}}$ *($\phi(t)$ is the Euler's ϕ function) and* $l = \begin{cases} \frac{3}{4}2^{\frac{n}{2}} \text{ if } n \text{ is even} \\ 2^{\frac{n-1}{2}} \text{ if } n \text{ is odd .} \end{cases}$.

Proof. We have $D_n = \{\sigma_n^1, \sigma_n^2, \ldots, \sigma_n^{n-1}, \sigma_n^n, \omega_n^1, \omega_n^2, \ldots, \omega_n^{n-1}, \omega_n^n\}$ acting on $V_n = \{0,1\}^n$. If a permutation $P \in D_n$ is such that $P(x) = x$ for some $x \in V_n$, then the components of x which belong to the same orbit must have the same value 0 or 1. Thus if there are $c_1, c_2, \ldots, c_j$ many cycles of different lengths, then the number of fixed points in V_n induced by P is $2^{c_1+\ldots+c_j}$.

First we find the number of fixed points induced by each σ_n^i, $1 \le i \le n$. The number of permutation cycles of σ_n^i is $gcd(n,i)$ each of length $\frac{n}{gcd(n,i)}$. Then the number of fixed points induced by σ_n^i is $2^{gcd(n,i)}$. Thus the total number of fixed points induced by σ_n^i for all $i(1 \le i \le n)$ is $\sum_{i=1}^{n} 2^{gcd(n,i)} = \sum_{t|n}\sum_{i,gcd(n,i)}^{n} 2^t = \sum_{t|n} 2^t \sum_{j,gcd(\frac{n}{t},j)=1} 1 = \sum_{t|n}\phi(\frac{n}{t})2^t$.

Let us now count the fixed points of V_n induced by the permutations w_n^i, $1 \le i \le n$. For an $i(1 \le i \le n)$, we have $\omega_n^i = \begin{pmatrix} 1 & 2 & 3 & \ldots & n \\ 1+i & n+i & n-1+i & \ldots & 2+i \end{pmatrix}$. In this permutation $1+i \to n-(1+i-2)+i$, i.e., $1+i \to n+1$, i.e., $1+i \to 1$ for $i \ne n$; further for $i = n$, $1 \to 1$. Again, for any k, $1 < k \le n$, $k \to n-k+2+i$. When $k \ne n-k+2+i \bmod n$, $n-k+2+i \to n-(n-k+2+i-2)+i$, i.e., $n-k+2+i \to k$. Thus, for all k, $1 \le k \le n$, there exists cycle of length at most 2. Let us now count the number of 1-cycle and 2-cycle for each ω_n^i, $1 \le i \le n$. k will form a 1-cycle which is (k) iff $k = n-k+2+i \bmod n$, i.e., $2k = i+2 \bmod n$, otherwise it will form a 2-cycle $(k, n-k+2+i \bmod n)$.

CASE 1A. n odd, i odd.
Let $i+2 = tn+r$, $1 \le r \le n$. If r is odd, $k = \frac{n+r}{2}$. On the other hand if r is even $k = \frac{r}{2}$.

CASE 1B. n odd and i even.
Since, $gcd(2,n) = 1$ and $i+2$ is even, $2k = i+2 \bmod n$ implies $k = \frac{i+2}{2} \bmod n$.

So for odd n there is exactly one 1-cycle and therefore number of 2-cycle is $\frac{n-1}{2}$. Thus the number of fixed points induced by each of ω_n^i is $2^1 \cdot 2^{\frac{n-1}{2}}$, i.e., $2^{\frac{n+1}{2}}$.

CASE 2A. n even and i odd.
Note that $i+2$ is odd. So n being even, it always leaves an odd

remainder after dividing $i+2$. Therefore, the relation $2k = i+2 \mod n$ is not possible at all.

CASE 2B. n even and i even.
Let $i+2 = tn+r$, $1 \leq r \leq n$. Note that r is always even as $i+2$ even. Therefore, the relation $2k = i+2 \mod n$ has two solutions which are $k = \frac{r}{2}$ and $k = \frac{n+r}{2}$.

So for even n, when i is odd there is no 1-cycle, that means the number of 2-cycle is $\frac{n}{2}$ and when i is even the number of 1-cycle is exactly 2 and the number of 2-cycle is $\frac{n-2}{2}$. Therefore, for odd i the total number of fixed points induced by ω_n^i is $2^{\frac{n}{2}}$ and $2^2 \cdot 2^{\frac{n-2}{2}} = 2^{\frac{n+2}{2}}$ otherwise.

Hence, using Burnside's Lemma we get,

$$\begin{aligned} d_n &= \tfrac{1}{2n}\textstyle\sum_{t|n}\phi(t)2^{\frac{n}{t}} + \tfrac{1}{2n}\begin{cases} \frac{n}{2}2^{\frac{n}{2}} + \frac{n}{2}2^{\frac{n+2}{2}} \text{ if } n \text{ is even} \\ n2^{\frac{n+1}{2}} \text{ if } n \text{ is odd} , \end{cases} \\ &= \tfrac{1}{2n}\textstyle\sum_{t|n}\phi(t)2^{\frac{n}{t}} + \begin{cases} \frac{3}{4}2^{\frac{n}{2}} \text{ if } n \text{ is even} \\ 2^{\frac{n-1}{2}} \text{ if } n \text{ is odd} . \end{cases} \end{aligned}$$

□

Note that the number of orbits under the action of C_n is g_n [26]. With the help of Theorem 3.5, we enumerate the number d_n and compare it with g_n in Table 1.

n	1	2	3	4	5	6	7	8	9	10
g_n	2	3	4	6	8	14	20	36	60	108
d_n	2	3	4	6	8	13	18	30	46	78

n	11	12	13	14	15	16
g_n	188	352	632	1182	2192	4116
d_n	126	224	380	687	1224	2250

TABLE 1. Comparison between g_n and d_n

Remark 1. *Let us name the set of all orbits induced by the action of C_n by RS_n and the orbits induced by the action of D_n by DS_n. Some of the orbits in RS_n will also be orbits in DS_n, where as some will merge with another orbit. Let $O_\sigma(x)$ is the orbit generated by $x = (x_1, \ldots, x_n)$ in RS_n. If for some i, $\omega^i(x) \in O_\sigma(x)$ then $O_\sigma(x)$ will be an orbit in DS_n. Otherwise, the set $\{x, \sigma_n^1(x), \ldots, \sigma^{n-1}(x), \omega_n^1(x), \ldots, \omega_n^n(x)\}$ will be an orbit in DS_n.*

Choose two integers i and j such that $j = i+k, k > 0$. Now $\omega^i(x) = \sigma_n^i\tau_n(x)$ and $\omega^j(x) = \sigma_n^j\tau_n(x) = \sigma_n^k\sigma_n^i\tau_n(x)$ That means $\omega^j(x)$ is the effect of k-cyclic rotation of $\omega^i(x)$, i.e., the set $O_\omega(x) = \{\omega^q(x), q = 1, 2, \ldots, n\}$ is an orbit in RS_n generated by x. If $\omega^i(x) = \sigma^p(x)$ for some integer p, i.e., $\omega^i(x) \in O_\sigma(x)$ then all the members of $O_\omega(x)$ belong to the orbit $O_\sigma(x)$. Therefore in DS_n the orbit $O_\sigma(x)$ remains as it is in RS_n. On the other hand, if $\omega^i(x)$ does not belong to the orbit $O_\sigma(x)$ then none of the members of $O_\omega(x)$ belong to $O_\sigma(x)$, which means $O_\sigma(x) \cup O_\omega(x)$ is an orbit in DS_n. As for example the orbit generated by 000000011 *in RS_9 is an orbit in DS_9, whereas the orbit generated by* 000001011 *and* 000001101 *respectively merge with each other and form a single orbit in DS_9.*

Let us denote the orbit generated by x in DS_n by $G_n(x)$. There are d_n many orbits up to equivalence. Without any loss of generality an orbit can be represented by the lexicographically least element belonging to it. We call each representative element as the leader of the corresponding orbit. Thus there are d_n many such leaders which we denote as $\Lambda_{n,0}, \Lambda_{n,1}, \ldots, \Lambda_{n,d_n-1}$ in lexicographical order. Thus an n-variable DSBF f can be described by a d_n length bit string $f(\Lambda_{n,0}), \ldots, f(\Lambda_{n,d_n-1})$.

4. Walsh transform of DSBFs

In this section we present the results related to Walsh transform values of DSBFs. The proofs of the results are almost similar to the analysis for RSBFs as presented in [18, 19, 26, 28]. Still we present them for completeness.

The Walsh transform value of f at w is given by $W_f(w) = \sum_{x\in\{0,1\}^n}(-1)^{f(x)\oplus x\cdot w}$. If f is a DSBF, then

$$W_f(w) = \sum_{i=0}^{d_n-1}(-1)^{f(\Lambda_{n,i})}\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot w}.$$

Then we have the following proposition.

Proposition 4.1. *Let $w, z \in V_n$ such that $z \in G_n(w)$. If f is a DSBF, then $W_f(z) = W_f(w)$.*

Proof. First we prove that

$$\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot w}=\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot z}$$

Since, $z\in G_n(w)$, either $z=\sigma^k(w)$ or $z=\omega^j(w)$ for some k and j. When $z=\sigma^k(w)$, we have, $\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot w}=\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{\sigma^k(x)\cdot\sigma^k(w)}=\sum_{y\in G_n(\Lambda_{n,i})}(-1)^{y\cdot z}$ (take $y=\sigma^k(x)$) $=\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot z}$.

On the other hand, if $z=\omega^j(w)$, then similarly it can be proved that $\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot w}=\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot z}$.

Then we have,

$$\begin{aligned}W_f(w) &= \textstyle\sum_{i=0}^{d_n-1}(-1)^{f(\Lambda_{n,i})}\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot w}\\ &= \textstyle\sum_{i=0}^{d_n-1}(-1)^{f(\Lambda_{n,i})}\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot z}\\ &= W_f(z)\end{aligned}\quad .$$

□

Thus we see that the distribution of the Walsh Transform values of a DSBF can be described by d_n many values. So we have the following lemma.

Lemma 4.2. *Let ${}_n\mathcal{M}$ be a $d_n\times d_n$ matrix, where the (i,j)-th element is $\sum_{x\in G_n(\Lambda_{n,i})}(-1)^{x\cdot\Lambda_{n,j}}$. Then Walsh transform of f can be determined as $[(-1)^{f(\Lambda_{n,0})},\ldots,(-1)^{f(\Lambda_{n,d_n-1})}]\ {}_n\mathcal{M}$.*

Corollary 4.3. *Let f be an n-variable DSBF.*

(1) *Nonlinearity of f is given by*

$$nl(f)=2^{n-1}-\frac{1}{2}max_{\Lambda_{n,j},0\le j<d_n}|\sum_{i=0}^{d_n-1}(-1)^{f(\Lambda_{n,i})}\ {}_nM_{i,j}|.$$

(2) *f is balanced iff $\sum_{i=0}^{d_n-1}(-1)^{f(\Lambda_{n,i})}\ {}_nM_{i,0}=0$.*
(3) *f is m-order Correlation Immune (respectively m-resilient) iff*

$$\sum_{i=0}^{d_n-1}(-1)^{f(\Lambda_{n,i})}\ {}_nM_{i,j}=0,\ \textit{for } 1\ \textit{(respectively } 0\textit{)}\le wt(\Lambda_{i,j})\le m,$$

where $wt(x)$ denotes the weight of x.

(4) *f is bent iff* $\sum_{i=0}^{d_n-1}(-1)^{f(\Lambda_{n,i})}\ {}_nM_{i,j} = \pm 2^{\frac{1}{2}}$, *for* $0 \leq j \leq d_n - 1$.

4.1. Investigation of the matrix ${}_n\mathcal{M}$ for odd n

It is clear that the members of an orbit are of same weight. Let us consider the weight of an orbit to be the weight of an element in that orbit. Since, n is odd, if weight of $x \in V_n$ is odd, weight of its complement $\overline{x}$ is even. Thus $G_n(x) \neq G_n(\overline{x})$. So we see that odd weight orbits and even weight orbits form two distinct classes each of size $\frac{d_n}{2}$. Also number of elements in $G_n(x)$ is same as that of $G_n(\overline{x})$. This gives the hints that the matrix ${}_n\mathcal{M}$ can be permuted in such a way that it can be written as $\left[\begin{array}{c|c} {}_n\mathcal{S} & {}_n\mathcal{S} \\ \hline {}_n\mathcal{S} & -{}_n\mathcal{S} \end{array}\right]$, as shown in the case of RSBFs in [18], where ${}_n\mathcal{S}$ is a $\frac{d_n}{2} \times \frac{d_n}{2}$ matrix.

Proposition 4.4. *[18] Let $X = (x_1, \ldots, x_n) \in V_n$ and $Y = (y_1, \ldots, y_n) \in V_n$. If $wt(X)$ and $wt(Y)$ is an even number and if n is odd, then*

$$\bigoplus_{i=1}^{n}(x_i \wedge y_i) = \bigoplus_{i=1}^{n}(\overline{x}_i \wedge y_i) = \bigoplus_{i=1}^{n}(x_i and \wedge \overline{y}_i) = 1 \oplus \bigoplus_{i=1}^{n}(\overline{x} \wedge \overline{y}_i).$$

Theorem 4.5. *If n is odd, the matrix ${}_n\mathcal{M}$ can be permuted to ${}_n\mathcal{M}^\pi$ such that ${}_n\mathcal{M}^\pi$ is of the form*

$${}_n\mathcal{M}^\pi = \left[\begin{array}{c|c} {}_n\mathcal{S} & {}_n\mathcal{S} \\ \hline {}_n\mathcal{S} & -{}_n\mathcal{S} \end{array}\right],$$

where ${}_n\mathcal{S}$ is a $\frac{d_n}{2} \times \frac{d_n}{2}$ matrix.

Proof. In the matrix ${}_n\mathcal{M}$, both d_n rows and d_n columns correspond to the d_n orbits which are in lexicographical order. First permute the rows of ${}_n\mathcal{M}$ so that the first $\frac{d_n}{2}$ rows correspond to the orbits of even weights in lexicographical order and the second $\frac{d_n}{2}$ rows correspond to the complements of the first $\frac{d_n}{2}$ orbits in the same order. That means if the i-th ($i = 0, \ldots, \frac{d_n}{2} - 1$) row corresponds to the orbit representative by $\Lambda_{n,i}$, where $\Lambda_{n,i}$ is the i-th element in the lexicographical order, then $(\frac{d_n}{2} + i)$-th row corresponds to the orbit generated by $\overline{\Lambda}_{n,i}$. Same permutation is given to the columns also. Finally we get the permuted matrix ${}_n\mathcal{M}^\pi$.

For $0 \leq r, c \leq \frac{d_n}{2} - 1$ we have

$$
\begin{aligned}
{}_n\mathcal{M}^{\pi}_{r,c} &= \sum_{x\in G_n(\Lambda_{n,r})}(-1)^{x\cdot\Lambda_{n,c}} \\
&= \sum_{x\in G_n(\Lambda_{n,r}}(-1)^{\oplus_{i=1}^{n}(x\wedge\Lambda_{(n,c)_i})} \\
{}_n\mathcal{M}^{\pi}_{r,c+\frac{d_n}{2}} &= \sum_{x\in G_n(\Lambda_{n,r})}(-1)^{x\cdot\Lambda_{n,c+\frac{d_n}{2}}} \\
&= \sum_{x\in G_n(\Lambda_{n,r}}(-1)^{\oplus_{i=1}^{n}(x\wedge\overline{\Lambda}_{(n,c)_i})} \\
{}_n\mathcal{M}^{\pi}_{r+\frac{d_n}{2},c} &= \sum_{x\in G_n(\Lambda_{n,r+\frac{d_n}{2}})}(-1)^{x\cdot\Lambda_{n,c}} \\
&= \sum_{x\in G_n(\Lambda_{n,r}}(-1)^{\oplus_{i=1}^{n}(\overline{x}\wedge\Lambda_{(n,c)_i})} \\
{}_n\mathcal{M}^{\pi}_{r+\frac{d_n}{2},c+\frac{d_n}{2}} &= \sum_{x\in G_n(\Lambda_{n,r+\frac{d_n}{2}})}(-1)^{x\cdot\Lambda_{n,c+\frac{d_n}{2}}} \\
&= \sum_{x\in G_n(\Lambda_{n,r}}(-1)^{\oplus_{i=1}^{n}(\overline{x}\wedge\overline{\Lambda}_{(n,c)_i})}
\end{aligned}
$$

Since $wt(\Lambda_{n,i})$ is even for $0 \leq i \leq \frac{d_n}{2} - 1$, then Proposition 4.4 implies that ${}_n\mathcal{M}^{\pi}_{r,c} = {}_n\mathcal{M}^{\pi}_{r,c+\frac{d_n}{2}} = {}_n\mathcal{M}^{\pi}_{r+\frac{d_n}{2},c} = -{}_n\mathcal{M}^{\pi}_{r+\frac{d_n}{2},c+\frac{d_n}{2}}$.

Hence the proof. □

As the total number of the n-variable DSBFs is lesser than that of the RSBFs for higher values of n (Table 1), one may attempt exhaustive search to find functions with good cryptographic properties such as nonlinearity, correlation immunity, balancedness etc. In this direction the matrix ${}_n\mathcal{M}^{\pi}$ will be much helpful rather than the matrix ${}_n\mathcal{M}$. For a DSBF f Walsh transform is given by $W_f = [(-1)^{f(\Lambda_{n,0})}, \ldots, (-1)^{f(\lambda_{n,d_n-1})}]\ {}_n\mathcal{M}$. The computation effort is d_n^2. We permute the representative elements of the orbits as described in Theorem 4.5, i.e., the first $\frac{d_n}{2}$ representative elements are of even wight written in lexicographical order and the second $\frac{d_n}{2}$ elements are the representative of the orbits in which complements of the first $\frac{d_n}{2}$ representative elements belong to. We rewrite f as $f = f_e||f_o$, where the $\frac{d_n}{2}$ length string f_e denotes the outputs at the representative elements with even weight and the $\frac{d_n}{2}$ length string f_o denotes the outputs at the representative elements with odd weight. Then we can write, $W_f = (w_e + w_o)||(w_e - w_o)$, where, $w_e = (-1)^{f_e}{}_n\mathcal{S}$ and $w_o = (-1)^{f_o}{}_n\mathcal{S}$, clearly the computation effort will be $2(\frac{d_n}{2})^2 + d_n$ which is much lesser than d_n^2.

4.2. Highly nonlinear Boolean functions on odd number of variables

We have studied the recently discovered 9-variable Boolean functions [13, 14] having nonlinearity 241. In [14, Table 1], 189 many 9-variable RSBFs having nonlinearity 241 have been described. We checked that out of them 21 many functions belong to the DSBF class. Therefore, one may note that the 9 variable Boolean functions having nonlinearity 241 in the DSBF class are more dense than those in RSBF class. This shows that the DSBF class contains highly nonlinear functions with good density. These functions have the maximum absolute value 52 in the autocorrelation spectra.

Further we have checked the two Patterson-Wiedemann functions [20] and found that they do not belong to the DSBF class. It will be interesting to study if some affine equivalent of these functions may belong to the DSBF class.

5. Conclusion

We present initial results on Boolean functions which are invariant under the action of the dihedral group. We provide the count of such functions and further study the properties of their Walsh spectra. We find that there are 9-variable Boolean functions having nonlinearity 241 in this class. Further investigation in this class may provide more interesting results regarding the cryptographic and combinatorial properties of such Boolean functions.

References

[1] A. Canteaut and M. Videau. Symmetric Boolean Function. *IEEE Transaction On Information Theory*, Volume 51, 2791–2811, 2005.

[2] J. Clark, J. Jacob, S. Maitra and P. Stănică. Almost Boolean functions: The design of Boolean functions by spectral inversion. *Computational Intelligence*, pages 450–462, Volume 20, Number 3, 2004.

[3] T. W. Cusick and P. Stănică. Fast evaluation, weights and nonlinearity of rotation-symmetric functions. *Discrete Mathematics*, pages 289-301, vol 258, no 1-3, 2002.

[4] T. W. Cusick and Y. Li. k-th order symmetric SAC Boolean functions and bisecting binomial coefficients. *Discrete Applied Mathematics*, Volume 149, 73–86, 2005.

[5] D. K. Dalai, K. C. Gupta and S. Maitra. Results on algebraic immunity for cryptographically significant Boolean functions. In *INDOCRYPT 2004*, number 3348 in Lecture Notes in Computer Science, pages 92–106, Springer Verlag, December 2004.

[6] D. K. Dalai, S. Maitra and S. Sarkar. Results on rotation symmetric Bent functions. In *Second International Workshop on Boolean Functions: Cryptography and Applications, BFCA 06*, March 13–15, 2006, LIFAR, University of Rouen, France.

[7] E. Filiol and C. Fontaine. Highly nonlinear balanced Boolean functions with a good correlation-immunity. In *Advances in Cryptology - EUROCRYPT'98*, number 1403 in Lecture Notes in Computer Science, pages 475–488, Springer-Verlag, 1998.

[8] C. Fontaine. On Some Cosets of the First-Order Reed-Muller Code with High Minimum Weight. In *IEEE Transactions on Information Theory*, 45(4):1237–1243, 1999.

[9] J. von zur Gathen and J. R. Roche. Polynomials with Two Values. *Combinatorica*, Volume 17(3), 345–362, 1997.

[10] K. Gopalakrishnan, D. G. Hoffman and D. R. Stinson. A Note on a Conjecture Concerning Symmetric Resilient Functions. *Information Processing Letters*, Volume 47(3), 139–143, 1993.

[11] F. Harary. Graph Theory. *Addison-Wesley Publishing Company*, 1972.

[12] M. Hell, A. Maximov and S. Maitra. On efficient implementation of search strategy for rotation symmetric Boolean functions. In *Ninth International Workshop on Algebraic and Combinatoral Coding Theory, ACCT 2004*, Black Sea Coast, Bulgaria, June 19–25, 2004.

[13] S. Kavut, S. Maitra and M. D. Yücel. Search for Boolean Functions with Excellent Profiles in the Rotation Symmetric Class. To be published in *IEEE Transactions on Information Theory*, 2007. An earlier version of this paper is available under the title "There exist Boolean functions on n (odd) variables having nonlinearity $> 2^{n-1} - 2^{\frac{n-1}{2}}$ if and only if $n > 7$" at IACR eprint server, http://eprint.iacr.org/2006/181, 28 May, 2006.

[14] S. Kavut, S. Maitra S. Sarkar and M. D. Yücel. Enumeration of 9-variable Rotation Symmetric Boolean Functions having Nonlinearity > 240. In *INDOCRYPT - 2006*, Lecture Notes in Computer Science 4329, Springer-Verlag, pp 266–279, 2006.

[15] S. Maitra and P. Sarkar. Characterization of symmetric bent functions – An elementary proof. *Journal of Combinatorial Mathematics and Combinatorial Computing*, Volume 43, 227–230, 2002.

[16] S. Maitra and P. Sarkar. Maximum Nonlinearity of Symmetric Boolean Functions on Odd Number of Variables. *IEEE Transactions on Information Theory*, Volume 48(9), 2626–2630, 2002.

[17] C. J. Mitchell. Enumerating Boolean functions of cryptographic significance. *Journal of Cryptology*, Volume 2(3), 155–170, 1990.

[18] A. Maximov, M. Hell and S. Maitra. Plateaued rotation symmetric Boolean functions on odd number of variables. In *First Workshop on Boolean Functions: Cryptography and Applications, BFCA 05*, LIFAR, University of Rouen, France, March 7–9, 2005.

[19] A. Maximov. Classes of plateaued rotation symmetric Boolean functions under transformation of Walsh spectra. In *Workshop on Coding and Cryptography, WCC 2005*, IACR eprint server, no. 2004/354.

[20] N. J. Patterson and D. H. Wiedemann. The covering radius of the $(2^{15}, 16)$ Reed-Muller code is at least 16276. *IEEE Transactions on Information Theory*, IT-29(3):354–356, 1983. See also correction in IT-36(2):443, 1990.

[21] J. Pieprzyk and C. X. Qu. Fast hashing and rotation-symmetric functions. *Journal of Universal Computer Science*, pages 20-31, vol 5, no 1 (1999).

[22] F. S. Roberts. *Applied Combinatorics*. Prentice-Hall, Inc, Englewood Cliffs, New Jersey.

[23] P. Sarkar and S. Maitra. Balancedness and Correlation Immunity of Symmetric Boolean Functions. To be published in *Discrete Mathematics*. Available at: http://dx.doi.org/10.1016/j.disc.2006.08.008 [last accessed February 2, 2007].

[24] S. Sarkar and S. Maitra. Efficient search for symmetric Boolean functions under constraints on Walsh spectra values. In *Second International Workshop on Boolean Functions: Cryptography and Applications, BFCA 06*, March 13–15, 2006, LIFAR, University of Rouen, France.

[25] P. Savicky. On the Bent Boolean Functions that are Symmetric. *European Journal of Combinatorics*, Volume 15, 407–410, 1994.

[26] P. Stănică and S. Maitra. Rotation symmetric Boolean functions – count and cryptographic properties. In *R. C. Bose Centenary Symposium on Discrete Mathematics and Applications*, Electronic Notes in Discrete Mathematics, Electronics Notes in Discrete Mathematics, volume 15, pages 178-183, Elsevier, December 2002. Available at : http://www1.elsevier.com/gej-ng/31/29/24/75/23/show/Products/notes/index.htt.

[27] P. Stănică and S. Maitra. A constructive count of rotation symmetric functions. *Information Processing Letters*, 88:299–304, 2003.

[28] P. Stănică, S. Maitra and J. Clark. Results on rotation symmetric bent and correlation immune Boolean functions. *Fast Software Encryption Workshop (FSE 2004)*, New Delhi, INDIA, LNCS 3017, Springer Verlag, pages 161–177, 2004.

[29] Y. X. Yang and B. Guo. Further enumerating Boolean functions of cryptographic significance. *Journal of Cryptology*, Volume 8(3), 115–122, 1995.

[30] C. K. Wu and E. Dawson. Correlation Immunity and Resiliency of Symmetric Boolean Functions. *Theoretical Computer Science*, Volume 312, 321–335, 2004.

Boolean Functions: Cryptography and Applications BFCA'07
Fonctions Booléennes : Cryptographie & Applications

SOME CONSTRUCTIONS OF BENT FUNCTIONS OF $n+2$ VARIABLES FROM BENT FUNCTIONS OF n VARIABLES *

Joan-Josep Climent [1], Francisco J. García [2] and Verónica Requena [3]

Abstract. In this paper we present two properties of the minterms of n variables. Then, using minterms of n variables and minterms of two variables, we present some methods to construct iteratively new bent functions of $n+2$ variables from bent functions of n variables.

1. Introduction

At the present time, S-boxes are an essential component in block ciphers. The implementation of an S-box needs nonlinear Boolean functions to guarantee the cryptographic effectiveness in order to resist powerful methods of attack such as the differential cryptanalysis. For an even number of inputs, Boolean functions of maximum nonlinearity are bent functions. There is a great number of bent functions, but unfortunately, the properties of bent

* *This work was partially supported by Spanish grant MTM2005-05759. The work of the third author was also supported by a grant for research students from the Vicerectorat d'Investigació, Desenvolupament i Innovació of the Universitat d'Alacant.*

[1] Institut Universitari d'Investigació Informàtica. Departament de Ciència de la Computació i Intel·ligència Artificial. Universitat d'Alacant, Ap. correus 99, E-03080 Alacant, SPAIN. email: jcliment@dccia.ua.es

[2] Departament de Fonaments de l'Anàlisi Econòmica. Universitat d'Alacant, Ap. correus 99, E-03080 Alacant, SPAIN. email: francisco.garcia@ua.es

[3] Departament de Ciència de la Computació i Intel·ligència Artificial. Universitat d'Alacant, Ap. correus 99, E-03080 Alacant, SPAIN.
email: vrequena@dccia.ua.es

functions, their classification, and their number are not totally known yet.

There are two families of methods for the construction of good S-boxes for cryptographic applications.

The simplest method is the direct construction of truth tables. For that purpose, Boolean functions that satisfy the strict avalanche criterion are used. They are denominated bent permutations. Nevertheless, we are still very far from having an exhaustive general method for their construction.

Alternatively, there is a method consisting in obtaining nonlinear functions by random generation. However multidimensional functions with the highest nonlinearity are not frequent, so it becomes difficult to find them randomly. Recently they have been developed genetic algorithms that are effective, but only in some particular cases.

It is well-known how to construct one-to-one S-boxes such that any linear combination of the output functions is balanced. It is also well-known with the process by which such linear combinations become bent. What is still unknown is the means for constructing all the S-boxes which satisfy those properties. It is for this reason, the study of the properties of bent functions and the methods to construct them has received a very high attention in the last decades.

The origin of bent functions goes back to a theoretical article of McFarland [10] on sets of finite differences in finite groups. One year after, Dillon [5] in his doctoral thesis systematized and extended the ideas of McFarland, proving a great quantity of properties. For example, all bent function of $n > 2$ variables has degree at most $n/2$, there are bent functions with degree equal to $n/2$, and the only symmetrical bent functions are the quadratic ones, existing exactly four of that functions for each n. The name *bent* for these functions is due to Rothaus [12].

¿From the truth tables of bent functions and linear functions, it is possible to construct bent functions with a greater number of variables. But not all the bent functions in 6 variables can be obtained from bent functions and linear functions with a less number of variables, as proved Chang [3]. This does not mean that it is not interesting to construct bent functions from functions with a fewer number of variables, but it is not possible to generate *all* of them in this way. In fact, thanks to Canteaut and Charpin [1] we know two infinite families of bent functions in n variables

that cannot be obtained from bent functions of smaller number of variables. In that paper, the authors also describe how the way in which they can be constructed, from a bent function of n variables, Boolean functions of $n-1$ and $n-2$ variables with a very high nonlinearity.

Following a different strategy, Hou and Langevin [9] described how, from a well-known bent function, new bent functions can be obtained with the same number of variables.

Another way to analyze bent functions consists in exploring the properties of the algebraic structures on $\mathrm{GF}(2^n)$. See, for example, Carlet and Guillot [2] or Hou [6, 8]. By this procedure some authors have been able to determine all the cubic bent functions in 8 variables from the cubic bent functions in 6 variables (see [7]), to find some homogeneous bent functions of degree 3 and with 8 or 10 variables (see [4]), or to characterize the homogeneous functions of 6 variables and degree 3 that are bent and those that are balanced (see [11]). In [11], the authors also discuss why the homogeneous functions could be very useful to design hash functions.

The construction of families of particular bent functions is important for the following two reasons. On the one hand, there exists the practical necessity to have functions of maximum nonlinearity to implement ciphers. On the other hand, there are theoretical reasons to discover properties and to contrast conjectures.

The mentioned literature makes an intensive use of the representation of Boolean functions in polynomial form, in matrix form and in sequential form. Nevertheless, the classical concept of minterm, which is, by the way, directly related to the implementation of logic circuits and its complexity, has not been frequently used.

This paper addresses the practical purposes involved in the generation of bent functions using the representation of Boolean functions as a sum of minterms. We will use this concept to obtain, from bent functions of n variables, new bent functions of $n+2$ variables. We will also verify that all functions obtained in such manner are different.

The rest of the paper is organized as follows. Firstly, in Section 2 we introduce some basic definitions and notations that are used. In Section 3, we present two general methods to construct bent functions of $n+2$ variables starting with bent functions of n variables, along with others important results that are necessary to prove the main theorems.

2. Preliminaries

Consider the binary field $\mathbb{Z}_2$ with the addition modulo 2 (denoted by $\oplus$) and the multiplication modulo 2. For any positive integer n, it is well-known that $\mathbb{Z}_2^n$ is a linear space over $\mathbb{Z}_2$ with the addition $\oplus$ given by

$$\boldsymbol{a} \oplus \boldsymbol{b} = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n)$$

for $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ in $\mathbb{Z}_2^n$. Also, we consider the inner product

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = a_1 b_1 \oplus a_2 b_2 \oplus \cdots \oplus a_n b_n$$

of $\boldsymbol{a}$ and $\boldsymbol{b}$. Furthermore, we say that $\boldsymbol{a} < \boldsymbol{b}$ if there exists k (with $1 \leq k \leq n$) such that

$$a_1 = b_1,\ a_2 = b_2,\ \ldots,\ a_{k-1} = b_{k-1} \quad \text{but} \quad a_k = 0 \text{ and } b_k = 1.$$

So we can order the elements $\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{2^n-1}$ of $\mathbb{Z}_2^n$ such that

$$\boldsymbol{e}_0 < \boldsymbol{e}_1 < \cdots < \boldsymbol{e}_{2^n-1}.$$

Finally, if $\boldsymbol{e}_i = (e_1^{(i)}, e_2^{(i)}, \ldots, e_{n-1}^{(i)}, e_n^{(i)}) \in \mathbb{Z}_2^n$, then

$$e_1^{(i)} 2^{n-1} + e_2^{(i)} 2^{n-2} + \cdots + e_{n-1}^{(i)} 2^1 + e_n^{(i)} 2^0 = i$$

and we call $\boldsymbol{e}_i$ the binary expansion of i. With this representation, we can identify the vector $\boldsymbol{e}_i$ with the integer i, and consequently, we can identify $\mathbb{Z}_2^n$ with $\mathbb{Z}_{2^n}$.

A Boolean function of n variables is a mapping $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$. We denote by $\mathcal{B}_n$ the set of Boolean functions of n variables. $\mathcal{B}_n$ is also a linear space over $\mathbb{Z}_2$ with the addition $\oplus$ given by

$$(f \oplus g)(\boldsymbol{x}) = f(\boldsymbol{x}) \oplus g(\boldsymbol{x})$$

for $f, g \in \mathcal{B}_n$. For a function f of $\mathcal{B}_n$, the $(0,1)$-sequence of length 2^n

$$\boldsymbol{\xi}_f = (f(\boldsymbol{e}_0),\ f(\boldsymbol{e}_1),\ \ldots,\ f(\boldsymbol{e}_{2^n-1}))$$

is called the **truth table** of f.

The truth table of a Boolean function can be obtained by its minterms. A **minterm** on n variables $x_1, x_2, \ldots, x_n$ is an expression of the form

$$m_{(u_1,u_2,\ldots,u_n)}(x_1, x_2, \ldots, x_n) = (\bar{u}_1 \oplus x_1)(\bar{u}_2 \oplus x_2) \cdots (\bar{u}_n \oplus x_n)$$

with $\bar{u} = 1 \oplus u$ for all $u \in \mathbb{Z}_2$. Now, for $i = 0, 1, 2, \ldots, 2^n - 1$, it is clear that $m_{\boldsymbol{e}_i}(\boldsymbol{x}) = 1$ if and only if $\boldsymbol{x} = \boldsymbol{e}_i$. We will write $m_i(\boldsymbol{x})$

instead of $m_{\boldsymbol{e}_i}(\boldsymbol{x})$. So, the truth table

$$(m_i(\boldsymbol{e}_0),\ m_i(\boldsymbol{e}_1),\ \ldots,\ m_i(\boldsymbol{e}_{2^n-1}))$$

of $m_i(\boldsymbol{x})$ has a 1 in the ith position and 0 elsewhere. Consequently,

$$\bigoplus_{i=0}^{2^n-1} m_i(\boldsymbol{x}) = 1 \tag{1}$$

and the set $\{m_0, m_1, m_2, \ldots, m_{2^n-1}\}$ is a basis for $\mathcal{B}_n$. So, any $f \in \mathcal{B}_n$ can be expressed as

$$f(\boldsymbol{x}) = \bigoplus_{i=0}^{2^n-1} f_i\ m_i(\boldsymbol{x})$$

where

$$f_i = \begin{cases} 1, & \text{if } i \in I, \\ 0, & \text{if } i \notin I \end{cases}$$

for a subset I of $\{0, 1, \ldots, 2^n - 1\}$ called the **support** of f and defined as

$$I = \{\boldsymbol{a} \in \mathbb{Z}_2^n \mid f(\boldsymbol{a}) = 1\}.$$

So,

$$f(\boldsymbol{x}) = \bigoplus_{i \in I} m_i(\boldsymbol{x})$$

and, therefore, I is the set of minterms of f.

The **Hamming weight** of a $(0,1)$-sequence $\boldsymbol{\alpha}$, denoted by $w(\boldsymbol{\alpha})$, is the number of 1s in $\boldsymbol{\alpha}$. The **Hamming distance** between two $(0,1)$-sequences $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, denoted by $d(\boldsymbol{\alpha}, \boldsymbol{\beta})$, is the number of positions where the two sequences differ. It is well-know that $d(\boldsymbol{\alpha}, \boldsymbol{\beta}) = w(\boldsymbol{\alpha} \oplus \boldsymbol{\beta})$.

Let f be a Boolean function and $\boldsymbol{\xi}_f$ its truth table. The **Hamming weight** of f, denoted by $w(f)$, is the Hamming weight of $\boldsymbol{\xi}_f$; that is $w(f) = w(\boldsymbol{\xi}_f)$, and therefore, $w(f)$ is the number of minterms in the expression of $f(\boldsymbol{x})$ as a sum of minterms.

Let f and g be two functions of $\mathcal{B}_n$ and $\boldsymbol{\xi}_f$ and $\boldsymbol{\xi}_g$ be the corresponding truth tables. The **Hamming distance** between f and g, denoted by $d(f, g)$, is the Hamming distance between $\boldsymbol{\xi}_f$ and $\boldsymbol{\xi}_g$; that is, $d(f, g) = d(\boldsymbol{\xi}_f, \boldsymbol{\xi}_g)$.

A $(0,1)$-sequence is **balanced** if it contains an equal number of 0s and 1s, so a function f in $\mathcal{B}_n$ is **balanced** if its truth table is balanced.

We say that f in $\mathcal{B}_n$ is an **affine function** if it takes the form

$$f(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x} \rangle \oplus b$$

where $\boldsymbol{a} \in \mathbb{Z}_2^n$ and $b \in \mathbb{Z}_2$. If $b = 0$, f is called a **linear function**. The set of affine functions is denoted by $\mathcal{A}_n$.

The **nonlinearity** of a function f in $\mathcal{B}_n$ is defined as

$$\mathrm{NL}(f) = \min\{d(f, \varphi) \mid \varphi \in \mathcal{A}_n\}$$

which is upper bounded (see [14]) by

$$\mathrm{NL}(f) \leq 2^{n-1} - 2^{\frac{n}{2}-1}.$$

The Boolean functions that achieve the maximum nonlinearity are called **bent functions** (see [14]). As a consequence, bent functions only exist for n even.

The following result (see [13, 14]), that we quote for further references, gives us a characterization of a bent function.

Theorem 2.1. *Let f be a function in $\mathcal{B}_n$. The following statements are equivalent.*

(1) *f is a bent function.*
(2) *For any $\boldsymbol{a} \in \mathbb{Z}_2^n \setminus \{\mathbf{0}\}$ the Boolean function $g_{\boldsymbol{a}}(\boldsymbol{x}) = f(\boldsymbol{x}) \oplus f(\boldsymbol{a} \oplus \boldsymbol{x})$ is balanced.*
(3) *For any $\boldsymbol{a} \in \mathbb{Z}_2^n$ the number of 1s in the truth table of the Boolean function $h_{\boldsymbol{a}}(\boldsymbol{x}) = f(\boldsymbol{x}) \oplus \langle \boldsymbol{a}, \boldsymbol{x} \rangle$ is $2^{n-1} \pm 2^{\frac{n}{2}-1}$.*

As a consequence of the previous theorem, the number of 1s in the truth table of a bent function f of n variables is $2^{n-1} \pm 2^{\frac{n}{2}-1}$; so that f is not balanced. Equivalently, the number of minterms in the expression of f as a sum of minterms is $2^{n-1} \pm 2^{\frac{n}{2}-1}$.

Finally, it is well-known that for any bent function $f(\boldsymbol{x})$, the functions $1 \oplus f(\boldsymbol{x})$ and $f(\boldsymbol{a} \oplus \boldsymbol{x})$ for all $\boldsymbol{a} \in \mathbb{Z}_2^n$, are also bent functions.

3. Main results

In the rest of the paper, we consider that $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is a vector of $\mathbb{Z}_2^n$ and that $\boldsymbol{y} = (y_1, y_2)$ is a vector of $\mathbb{Z}_2^2$.

Firstly, we introduce two important properties of the minterms which allow us to construct functions of $n+2$ variables from functions of n variables. In fact, we convert the minterms of a function of n variables in minterms of a function of $n+2$ variables.

Lemma 3.1. *For each minterm of n variables, it is possible to construct four different minterms of $n+2$ variables.*

Proof. Consider a minterm $m_a(\boldsymbol{x})$ of n variables, and consider also the four minterms in 2 variables

$$m_b(\boldsymbol{y}) \quad \text{for} \quad b = 0, 1, 2, 3.$$

Assume that $(a_1, a_2, \ldots, a_n)$ and (b_1, b_2) are the binary expansion of the integers a and b, respectively. Then

$$\begin{aligned} m_b(\boldsymbol{y})\ m_a(\boldsymbol{x}) &= m_{(b_1,b_2)}(\boldsymbol{y})\ m_{(a_1,a_2,\ldots,a_n)}(\boldsymbol{x}) \\ &= (\bar{b}_1 \oplus y_1)(\bar{b}_2 \oplus y_2)(\bar{a}_1 \oplus x_1)(\bar{a}_2 \oplus x_2)\cdots(\bar{a}_n \oplus x_n) \\ &= m_{(b_1,b_2,a_1,a_2,\ldots,a_n)}(\boldsymbol{y}, \boldsymbol{x}) \\ &= m_c(\boldsymbol{y}, \boldsymbol{x}) \end{aligned}$$

where $c = b_1 2^{n+1} + b_2 2^n + a$. □

The proof of the previous lemma tell us that the four minterms of $n+2$ variables that can be obtained starting from the minterm $m_a(\boldsymbol{x})$ of n variables are

$$m_a(\boldsymbol{y}, \boldsymbol{x}),\ m_{2^n+a}(\boldsymbol{y}, \boldsymbol{x}),\ m_{2^{n+1}+a}(\boldsymbol{y}, \boldsymbol{x}),\ \text{and}\ m_{2^n+2^{n+1}+a}(\boldsymbol{y}, \boldsymbol{x}).$$

Furthermore, minterms have the following property that make them operative from the algebraic point of view.

Lemma 3.2. $m_{\boldsymbol{a}}(\boldsymbol{b} \oplus \boldsymbol{x}) = m_{\boldsymbol{a}\oplus\boldsymbol{b}}(\boldsymbol{x})$ *for all* $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_2^n$.

Proof. Assume that

$$\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \qquad \text{and} \qquad \boldsymbol{b} = (b_1, b_2, \ldots, b_n)$$

then

$$\begin{aligned} m_{\boldsymbol{a}}(\boldsymbol{b} \oplus \boldsymbol{x}) &= (\bar{a}_1 \oplus b_1 \oplus x_1)(\bar{a}_2 \oplus b_2 \oplus x_2)\cdots(\bar{a}_n \oplus b_n \oplus x_n) \\ &= (\overline{a_1 \oplus b_1} \oplus x_1)(\overline{a_2 \oplus b_2} \oplus x_2)\cdots(\overline{a_n \oplus b_n} \oplus x_n) \\ &= m_{\boldsymbol{a}\oplus\boldsymbol{b}}(\boldsymbol{x}) \end{aligned}$$

because $\bar{a}_i \oplus b_i = 1 \oplus a_i \oplus b_i = \overline{a_i \oplus b_i}$ for $i = 1, 2, \ldots, n$. □

The following theorem is one of the main results of this paper. Here, we describe an iterative method to construct bent functions of $n+2$ variables, starting with two bent function of n variables.

y_1	y_2	$\boldsymbol{x}$	$m_0(\boldsymbol{y})$	$m_1(\boldsymbol{y})$	$m_2(\boldsymbol{y})$	$m_3(\boldsymbol{y})$	$B_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})$
0	**0**	$\boldsymbol{\tau}$	**1**	**0**	**0**	**0**	$\boldsymbol{\xi}_0 \oplus \boldsymbol{\Lambda_a}$
0	**1**	$\boldsymbol{\tau}$	**0**	**1**	**0**	**0**	$\boldsymbol{\xi}_0 \oplus b_2\mathbf{1} \oplus \boldsymbol{\Lambda_a}$
1	**0**	$\boldsymbol{\tau}$	**0**	**0**	**1**	**0**	$\boldsymbol{\xi}_1 \oplus b_1\mathbf{1} \oplus \boldsymbol{\Lambda_a}$
1	**1**	$\boldsymbol{\tau}$	**0**	**0**	**0**	**1**	$\mathbf{1} \oplus \boldsymbol{\xi}_1 \oplus b_1\mathbf{1} \oplus b_2\mathbf{1} \oplus \boldsymbol{\Lambda_a}$

TABLE 1. Truth table of $B_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})$

Theorem 3.3. *Let $f_0(\boldsymbol{x})$ and $f_1(\boldsymbol{x})$ be bent functions of n variables and assume that (i_0, i_1, i_2, i_3) is a permutation of $(0, 1, 2, 3)$. Then*

$$B(\boldsymbol{y},\boldsymbol{x}) = (m_{i_0}(\boldsymbol{y}) \oplus m_{i_1}(\boldsymbol{y}))\, f_0(\boldsymbol{x}) \oplus m_{i_2}(\boldsymbol{y}) f_1(\boldsymbol{x}) \oplus m_{i_3}(\boldsymbol{y})(1 \oplus f_1(\boldsymbol{x}))$$

is a bent function of $n + 2$ variables.

Proof. According to Theorem 2.1 we must prove that the number of 1s of the truth table (that is, the number of minterms) of the Boolean function

$$B_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x}) = B(\boldsymbol{y},\boldsymbol{x}) \oplus \langle(\boldsymbol{b},\boldsymbol{a}),(\boldsymbol{y},\boldsymbol{x})\rangle$$

is $2^{n+1} \pm 2^{\frac{n}{2}}$ for all $(\boldsymbol{b},\boldsymbol{a}) \in \mathbb{Z}_2^2 \times \mathbb{Z}_2^n$.

Let us assume that $(i_0, i_1, i_2, i_3) = (0, 1, 2, 3)$. Then

$$B_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x}) = (m_0(\boldsymbol{y}) \oplus m_1(\boldsymbol{y}))\, f_0(\boldsymbol{x}) \oplus m_2(\boldsymbol{y}) f_1(\boldsymbol{x}) \oplus m_3(\boldsymbol{y})(1 \oplus f_1(\boldsymbol{x})) \oplus b_1 y_1 \oplus b_2 y_2 \oplus \langle \boldsymbol{a},\boldsymbol{x}\rangle$$

where $\boldsymbol{b} = (b_1, b_2)$.

So, if $\mathbf{0}$ and $\mathbf{1}$ are the $2^n \times 1$ arrays with all entries equal to 0 and 1 respectively; $\boldsymbol{\tau}$ is the $2^n \times n$ array whose ith row is $\boldsymbol{e}_i$; $\boldsymbol{\xi}_0$ and $\boldsymbol{\xi}_1$ are the truth table of f_0 and f_1 respectively; and $\boldsymbol{\Lambda_a}$ is the truth table of the linear function $\langle \boldsymbol{a},\boldsymbol{x}\rangle$, then the last column of Table 1 shows the truth table of $B_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})$. Now, each column of Table 2 represents the four blocks of the truth table of $B_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})$ for the different values of $\boldsymbol{b}$.

Since f_0 and f_1 are bent functions, from Theorem 2.1, we have that, for $j = 0, 1$, the number of 1s of $\boldsymbol{\xi}_j \oplus \boldsymbol{\Lambda_a}$ is $2^{n-1} \pm 2^{\frac{n}{2}-1}$, and therefore, the number of 1s of $\boldsymbol{\xi}_j \oplus \mathbf{1} \oplus \boldsymbol{\Lambda_a}$ is $2^{n-1} \mp 2^{\frac{n}{2}-1}$. So, in any case, we have three blocks in which the number of 1s is $2^{n-1} + 2^{\frac{n}{2}-1}$

$b_1=0 \quad b_2=0$	$b_1=0 \quad b_2=1$	$b_1=1 \quad b_2=0$	$b_1=1 \quad b_2=1$
$\boldsymbol{\xi}_0 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_0 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_0 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_0 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$
$\boldsymbol{\xi}_0 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_0 \oplus \mathbf{1} \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_0 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_0 \oplus \mathbf{1} \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$
$\boldsymbol{\xi}_1 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_1 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_1 \oplus \mathbf{1} \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_1 \oplus \mathbf{1} \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$
$\mathbf{1} \oplus \boldsymbol{\xi}_1 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_1 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\boldsymbol{\xi}_1 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$	$\mathbf{1} \oplus \boldsymbol{\xi}_1 \oplus \boldsymbol{\Lambda}_{\boldsymbol{a}}$

TABLE 2. Truth table of $B_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})$ for the different values of $\boldsymbol{b}=(b_1,b_2)$

and one block in which the number of 1s is $2^{n-1}-2^{\frac{n}{2}-1}$, or three blocks in which the number of 1s is $2^{n-1}-2^{\frac{n}{2}-1}$ and one block in which the number of 1s is $2^{n-1}+2^{\frac{n}{2}-1}$. Consequently, the number of minterms of $B_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})$ is always $2^{n+1}+2^{\frac{n}{2}}$ or $2^{n+1}-2^{\frac{n}{2}}$.

Finally, if (i_0,i_1,i_2,i_3) is a permutation of $(0,1,2,3)$ other than $(0,1,2,3)$, then the four blocks of the truth table of $B_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})$ given in Table 2 are permuted according to (i_0,i_1,i_2,i_3) and therefore, the same result follows. □

Observe that as a consequence of Lemma 3.1 and Theorem 3.3 if I_0 and I_1 are the set of minterms of $f_0(\boldsymbol{x})$ and $f_1(\boldsymbol{x})$, respectively, and if (a_0,a_1,a_2,a_3) is a permutation of $(0,2^n,2^{n+1},2^n+2^{n+1})$, then the set of minterms of the bent function $B(\boldsymbol{y},\boldsymbol{x})$ constructed in Theorem 3.3 is

$$\begin{aligned} I = &\{a_0+a, a_1+a \mid a\in I_0\} \cup \{a_2+a \mid a\in I_1\} \\ &\cup \{a_3+a \mid a\in \bar{I}_1\} \end{aligned} \tag{2}$$

where $\bar{I}_1=\mathbb{Z}_{2^n}\setminus I_1$ is the set of minterms of $1\oplus f_1(\boldsymbol{x})$.

Now, from expression (1) we have that

$$m_0(\boldsymbol{y})\oplus m_1(\boldsymbol{y})\oplus m_2(\boldsymbol{y})\oplus m_3(\boldsymbol{y})=1. \tag{3}$$

So, if we take $f_0(\boldsymbol{x})=f_1(\boldsymbol{x})=f(\boldsymbol{x})$ in the previous theorem, we have the following result.

Corollary 3.4. *If $f(\boldsymbol{x})$ is a bent function of n variables and if $i\in\{0,1,2,3\}$, then*

$$F(\boldsymbol{y},\boldsymbol{x})=f(\boldsymbol{x})\oplus m_i(\boldsymbol{y})$$

is a bent function of $n+2$ variables.

Observe that, according to expression (2), the set of minterms of the bent function $F(\boldsymbol{y},\boldsymbol{x})$ constructed in the previous corollary is,

$$\{a_0 + a, a_1 + a, a_2 + a \mid a \in I\} \cup \left\{a_3 + a \mid a \in \bar{I}\right\}$$

where I is the set of minterms of $f(\boldsymbol{x})$ and $\bar{I} = \mathbb{Z}_{2^n} \setminus I$ is the set of minterms of $1 \oplus f(\boldsymbol{x})$.

On the other hand, remember that if $f(\boldsymbol{x})$ is a bent function of n variables and if $\boldsymbol{c}$ is a nonzero vector of $\mathbb{Z}_2^n$, then $f(\boldsymbol{c} \oplus \boldsymbol{x})$ is also a bent function with the same number of minterms that $f(\boldsymbol{x})$. So, if we take $f_0(\boldsymbol{x}) = f(\boldsymbol{x})$, and $f_1(\boldsymbol{x}) = f(\boldsymbol{c} \oplus \boldsymbol{x})$ then, we have the following result.

Corollary 3.5. *If $f(\boldsymbol{x})$ is a bent function of n variables, we consider $\boldsymbol{c} \in \mathbb{Z}_2^n$ with $\boldsymbol{c} \neq \boldsymbol{0}$, and if (i_0, i_1, i_2, i_3) is a permutation of $(0, 1, 2, 3)$, then*

$$\begin{aligned} G(\boldsymbol{y}, \boldsymbol{x}) = {} & (m_{i_0}(\boldsymbol{y}) \oplus m_{i_1}(\boldsymbol{y}))\, f(\boldsymbol{x}) \oplus m_{i_2}(\boldsymbol{y}) f(\boldsymbol{c} \oplus \boldsymbol{x}) \\ & \oplus m_{i_3}(\boldsymbol{y})\, (1 \oplus f(\boldsymbol{c} \oplus \boldsymbol{x})) \end{aligned}$$

is a bent function of $n + 2$ variables.

If I is the set of minterms of $f(\boldsymbol{x})$ and if $\boldsymbol{c}$ is the binary expansion of the integer c, then by Lemma 3.2, $c + I = \{c + a \mid a \in I\}$ is the set of minterms of $f(\boldsymbol{c} \oplus \boldsymbol{x})$. So, according to expressions (2) and (3), if (a_0, a_1, a_2, a_3) is a permutation of $(0, 2^n, 2^{n+1}, 2^n + 2^{n+1})$, then the set of minterms of the bent function $G(\boldsymbol{y}, \boldsymbol{x})$ constructed in the previous corollary is,

$$\{a_0 + a, a_1 + a, a_2 + c + a \mid a \in I\} \cup \left\{a_3 + c + a \mid a \in \bar{I}\right\}$$

where I is the set of minterms of $f(\boldsymbol{x})$.

Next, we present a new construction of a bent function of $n+2$ variables, from a bent function $f(\boldsymbol{x})$ of n variables and some shifts of $f(\boldsymbol{x})$, that can not be obtained from Theorem 3.3.

Theorem 3.6. *Let $f(\boldsymbol{x})$ be a bent function of n variables and consider $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_2^n \setminus \{\boldsymbol{0}\}$ such that*

$$f(\boldsymbol{x}) \oplus f(\boldsymbol{u} \oplus \boldsymbol{x}) \oplus f(\boldsymbol{v} \oplus \boldsymbol{x}) \oplus f(\boldsymbol{u} \oplus \boldsymbol{v} \oplus \boldsymbol{x}) = 1. \tag{4}$$

If (i_0, i_1, i_2, i_3) is a permutation of $(0, 1, 2, 3)$, then

$$\begin{aligned} H(\boldsymbol{y}, \boldsymbol{x}) = {} & m_{i_0}(\boldsymbol{y}) f(\boldsymbol{x}) \oplus m_{i_1}(\boldsymbol{y}) f(\boldsymbol{u} \oplus \boldsymbol{x}) \oplus m_{i_2}(\boldsymbol{y}) f(\boldsymbol{v} \oplus \boldsymbol{x}) \\ & \oplus m_{i_3}(\boldsymbol{y})\, (1 \oplus f(\boldsymbol{u} \oplus \boldsymbol{v} \oplus \boldsymbol{x})) \end{aligned}$$

is a bent function of $n + 2$ variables.

Proof. According to Theorem 2.1, we must prove that the function

$$H_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x}) = H(\boldsymbol{y},\boldsymbol{x}) \oplus H((\boldsymbol{b},\boldsymbol{a}) \oplus (\boldsymbol{y},\boldsymbol{x}))$$

is balanced for all nonzero vectors $(\boldsymbol{b},\boldsymbol{a}) \in \mathbb{Z}_2^2 \times \mathbb{Z}_2^n$. In the following, we identify the vector $\boldsymbol{b} \in \mathbb{Z}_2^2$ with its integer representation in $\mathbb{Z}_{2^2}$.

Firstly, observe that from expression (4) and Lemma 3.2, we have that

$$\begin{aligned} H_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x}) = {} & m_{i_0}(\boldsymbol{y})f(\boldsymbol{x}) \oplus m_{i_1}(\boldsymbol{y})f(\boldsymbol{u}\oplus\boldsymbol{x}) \oplus m_{i_2}(\boldsymbol{y})f(\boldsymbol{v}\oplus\boldsymbol{x}) \\ & \oplus m_{i_3}(\boldsymbol{y})\big(1 \oplus f(\boldsymbol{u}\oplus\boldsymbol{v}\oplus\boldsymbol{x})\big) \\ & \oplus m_{i_0\oplus\boldsymbol{b}}(\boldsymbol{y})f(\boldsymbol{a}\oplus\boldsymbol{x}) \oplus m_{i_1\oplus\boldsymbol{b}}(\boldsymbol{y})f(\boldsymbol{a}\oplus\boldsymbol{u}\oplus\boldsymbol{x}) \\ & \oplus m_{i_2\oplus\boldsymbol{b}}(\boldsymbol{y})f(\boldsymbol{a}\oplus\boldsymbol{v}\oplus\boldsymbol{x}) \\ & \oplus m_{i_3\oplus\boldsymbol{b}}(\boldsymbol{y})\big(1 \oplus f(\boldsymbol{a}\oplus\boldsymbol{u}\oplus\boldsymbol{v}\oplus\boldsymbol{x})\big). \end{aligned} \tag{5}$$

So, we consider different cases depending on the values of $(\boldsymbol{b},\boldsymbol{a})$.

Firstly, assume that $\boldsymbol{a} = \mathbf{0}_n$ and that $\boldsymbol{b} \neq \mathbf{0}_2$.

If $\boldsymbol{b} = 1$, using expression (1) we obtain, after some tedious algebraic manipulations, that

$$H_{(1,\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x}) = f(\boldsymbol{x}) \oplus f(\boldsymbol{u}\oplus\boldsymbol{x}).$$

Now, if $\boldsymbol{\xi}$ and $\boldsymbol{\xi}_{\boldsymbol{u}}$ are the truth table of $f(\boldsymbol{x})$ and $f(\boldsymbol{u}\oplus\boldsymbol{x})$, respectively, then the truth table of $H_{(1,\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})$ has four blocks,

$$\boldsymbol{\xi}\oplus\boldsymbol{\xi}_{\boldsymbol{u}} \qquad \boldsymbol{\xi}\oplus\boldsymbol{\xi}_{\boldsymbol{u}} \qquad \boldsymbol{\xi}\oplus\boldsymbol{\xi}_{\boldsymbol{u}} \qquad \boldsymbol{\xi}\oplus\boldsymbol{\xi}_{\boldsymbol{u}}$$

and therefore, it is balanced, because $\boldsymbol{\xi}\oplus\boldsymbol{\xi}_{\boldsymbol{u}}$ is balanced by Theorem 2.1 since $f(\boldsymbol{x})$ is a bent function.

A similar argument follows for $\boldsymbol{b} = 2$ and $\boldsymbol{b} = 3$ because

$$H_{(2,\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x}) = f(\boldsymbol{x}) \oplus f(\boldsymbol{v}\oplus\boldsymbol{x})$$

and

$$H_{(3,\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x}) = f(\boldsymbol{u}\oplus\boldsymbol{x}) \oplus f(\boldsymbol{v}\oplus\boldsymbol{x}).$$

Now, assume that $\boldsymbol{a} \neq \mathbf{0}_n$ and $\boldsymbol{b} = \mathbf{0}_2$.

Then, from expression (5) we have that

$$\begin{aligned} H_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x}) = {} & m_{i_0}(\boldsymbol{y})\big(f(\boldsymbol{x}) \oplus f(\boldsymbol{a}\oplus\boldsymbol{x})\big) \\ & \oplus m_{i_1}(\boldsymbol{y})\big(f(\boldsymbol{u}\oplus\boldsymbol{x}) \oplus f(\boldsymbol{a}\oplus\boldsymbol{u}\oplus\boldsymbol{x})\big) \end{aligned}$$

$$\oplus\, m_{i_2}(\boldsymbol{y})\big(f(\boldsymbol{v}\oplus\boldsymbol{x})\oplus f(\boldsymbol{a}\oplus\boldsymbol{v}\oplus\boldsymbol{x})\big)$$
$$\oplus\, m_{i_3}(\boldsymbol{y})\big(f(\boldsymbol{u}\oplus\boldsymbol{v}\oplus\boldsymbol{x})\oplus f(\boldsymbol{a}\oplus\boldsymbol{u}\oplus\boldsymbol{v}\oplus\boldsymbol{x})\big)$$

and taking into account that, by Theorem 2.1, the functions

$$\begin{array}{cc} f(\boldsymbol{x})\oplus f(\boldsymbol{a}\oplus\boldsymbol{x}), & f(\boldsymbol{u}\oplus\boldsymbol{x})\oplus f(\boldsymbol{a}\oplus\boldsymbol{u}\oplus\boldsymbol{x}),\\ f(\boldsymbol{d}\oplus\boldsymbol{x})\oplus f(\boldsymbol{a}\oplus\boldsymbol{v}\oplus\boldsymbol{x}), & f(\boldsymbol{u}\oplus\boldsymbol{v}\oplus\boldsymbol{x})\oplus f(\boldsymbol{a}\oplus\boldsymbol{u}\oplus\boldsymbol{v}\oplus\boldsymbol{x})\end{array}$$

are balanced because $f(\boldsymbol{x})$, $f(\boldsymbol{u}\oplus\boldsymbol{x})$, $f(\boldsymbol{v}\oplus\boldsymbol{x})$, and $f(\boldsymbol{u}\oplus\boldsymbol{v}\oplus\boldsymbol{x})$ are bent functions, we obtain that $H_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})$ is balanced.

Finally, assume that $\boldsymbol{a}\neq\mathbf{0}_n$ and $\boldsymbol{b}\neq\mathbf{0}_2$. Taking into account that $\oplus$ in $\mathbb{Z}_2^2=\{0,1,2,3\}$ is defined by the table

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

and that $\boldsymbol{b}\neq 0$, we have that

$$(i_0\oplus\boldsymbol{b},i_1\oplus\boldsymbol{b},i_2\oplus\boldsymbol{b},i_3\oplus\boldsymbol{b})=\begin{cases}(i_1,i_0,i_3,i_2), & \text{if } \boldsymbol{b}=1,\\ (i_2,i_3,i_0,i_1), & \text{if } \boldsymbol{b}=2,\\ (i_3,i_2,i_1,i_0), & \text{if } \boldsymbol{b}=3.\end{cases}$$

So, for $\boldsymbol{b}=1$, expression (5) can be written in the following way depending on the values of $\boldsymbol{a}$.

For $\boldsymbol{a}=\boldsymbol{u}$, we have that

$$H_{(1,\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})=m_{i_2}(\boldsymbol{y})\oplus m_{i_3}(\boldsymbol{y})$$

whose truth table has four blocks

$$\mathbf{0}\qquad\mathbf{0}\qquad\mathbf{1}\qquad\mathbf{1}$$

not necessarily in this order, and consequently it is balanced because the length of each block is 2^n.

For $\boldsymbol{a}=\boldsymbol{v}$, we have that

$$\begin{aligned}H_{(1,\boldsymbol{a})}(\boldsymbol{y},\boldsymbol{x})=\ & m_{i_0}(\boldsymbol{y})\big(f(\boldsymbol{x})\oplus f(\boldsymbol{a}\oplus\boldsymbol{u}\oplus\boldsymbol{x})\big)\\ & \oplus\, m_{i_1}(\boldsymbol{y})\big(f(\boldsymbol{u}\oplus\boldsymbol{x})\oplus f(\boldsymbol{a}\oplus\boldsymbol{x})\big)\\ & \oplus\, m_{i_2}(\boldsymbol{y})\big(f(\boldsymbol{a}\oplus\boldsymbol{x})\oplus 1\oplus f(\boldsymbol{u}\oplus\boldsymbol{x})\big)\end{aligned}$$

$$\oplus\, m_{i_3}(\boldsymbol{y})\big(1 \oplus f(\boldsymbol{a} \oplus \boldsymbol{u} \oplus \boldsymbol{x}) \oplus f(\boldsymbol{x})\big)$$

which is balanced because $\boldsymbol{a} \neq \boldsymbol{u}$ and each one of the functions that multiplies a minterm is balanced.

For $\boldsymbol{a} \neq \boldsymbol{u}$ and $\boldsymbol{a} \neq \boldsymbol{v}$, we have by a similar argument that $H_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y}, \boldsymbol{x})$ is balanced.

Finally, for $\boldsymbol{b} = 2$ and $\boldsymbol{b} = 3$ we can use the same argument to obtain that $H_{(\boldsymbol{b},\boldsymbol{a})}(\boldsymbol{y}, \boldsymbol{x})$ is balanced. □

Observe that if I is the set of minterms of $f(\boldsymbol{x})$, then $u + I$, $v + I$ and $w + I$ are the sets of minterms of $f(\boldsymbol{u} \oplus \boldsymbol{x})$, $f(\boldsymbol{v} \oplus \boldsymbol{x})$, and $f(\boldsymbol{u} \oplus \boldsymbol{v} \oplus \boldsymbol{x})$, respectively, where w is the integer whose binary expansion is $\boldsymbol{u} \oplus \boldsymbol{v}$. So, expression (4) tells us that

$$I \cup (u + I) \cup (v + I) \cup (w + I) = \mathbb{Z}_{2^n}.$$

Furthermore, if (a_0, a_1, a_2, a_3) is a permutation of $(0, 2^n, 2^{n+1}, 2^n + 2^{n+1})$, then, the set of minterms of the bent function $H(\boldsymbol{y}, \boldsymbol{x})$ constructed in the previous theorem is,

$$\{a_0 + a, a_1 + u + a, a_2 + v + a \mid a \in I\} \cup \left\{a_3 + w + a \mid a \in \bar{I}\right\}.$$

One question that arises at this point is the following: starting with the same bent function $f(\boldsymbol{x})$ of n variables, the bent functions $F(\boldsymbol{y}, \boldsymbol{x})$, $G(\boldsymbol{y}, \boldsymbol{x})$, and $H(\boldsymbol{y}, \boldsymbol{x})$, in $n + 2$ variables, constructed following Corollaries 3.4 and 3.5, and Theorem 3.6 are will they be different? And indeed they are and the following theorem will prove it.

Theorem 3.7. *Let $f(\boldsymbol{x})$ be a bent function of n variables. Let (i_0, i_1, i_2, i_3), (j_0, j_1, j_2, j_3), and (k_0, k_1, k_2, k_3) be permutations of $(0, 1, 2, 3)$. Assume that $F(\boldsymbol{y}, \boldsymbol{x})$, $G(\boldsymbol{y}, \boldsymbol{x})$, and $H(\boldsymbol{y}, \boldsymbol{x})$ are the functions constructed in Corollaries 3.4, 3.5, and Theorem 3.6 using permutations (i_0, i_1, i_2, i_3), (j_0, j_1, j_2, j_3), and (k_0, k_1, k_2, k_3) respectively. Then $F(\boldsymbol{y}, \boldsymbol{x}) \neq G(\boldsymbol{y}, \boldsymbol{x})$, $F(\boldsymbol{y}, \boldsymbol{x}) \neq H(\boldsymbol{y}, \boldsymbol{x})$ and $G(\boldsymbol{y}, \boldsymbol{x}) \neq H(\boldsymbol{y}, \boldsymbol{x})$.*

Proof. According to Corollaries 3.4, 3.5, and Theorem 3.6 we have that

$$\begin{aligned} F(\boldsymbol{y}, \boldsymbol{x}) &= f(\boldsymbol{x}) \oplus m_{i_3}(\boldsymbol{y}) \\ G(\boldsymbol{y}, \boldsymbol{x}) &= \left(m_{j_0}(\boldsymbol{y}) \oplus m_{j_1}(\boldsymbol{y})\right) f(\boldsymbol{x}) \oplus m_{j_2}(\boldsymbol{y}) f(\boldsymbol{c} \oplus \boldsymbol{x}) \\ &\quad \oplus m_{j_3}(\boldsymbol{y})\left(1 \oplus f(\boldsymbol{c} \oplus \boldsymbol{x})\right) \end{aligned}$$

$$H(\boldsymbol{y},\boldsymbol{x}) = m_{k_0}(\boldsymbol{y})f(\boldsymbol{x}) \oplus m_{k_1}(\boldsymbol{y})f(\boldsymbol{u}\oplus\boldsymbol{x}) \oplus m_{k_2}(\boldsymbol{y})f(\boldsymbol{v}\oplus\boldsymbol{x}) \oplus m_{k_3}(\boldsymbol{y})\,(1\oplus f(\boldsymbol{u}\oplus\boldsymbol{v}\oplus\boldsymbol{x}))$$

If $\boldsymbol{\xi}$ is the truth table of $f(\boldsymbol{x})$, then the truth table of $F(\boldsymbol{y},\boldsymbol{x})$, $G(\boldsymbol{y},\boldsymbol{x})$, and $H(\boldsymbol{y},\boldsymbol{x})$ have four blocks (not necessarily in that order and not the same order for all):

$$\begin{array}{lcccc} F: & \boldsymbol{\xi} & \boldsymbol{\xi} & \boldsymbol{\xi} & \boldsymbol{\xi}\oplus\mathbf{1} \\ G: & \boldsymbol{\xi} & \boldsymbol{\xi} & \boldsymbol{\xi}_{\boldsymbol{c}} & \mathbf{1}\oplus\boldsymbol{\xi}_{\boldsymbol{c}} \\ H: & \boldsymbol{\xi} & \boldsymbol{\xi}_{\boldsymbol{u}} & \boldsymbol{\xi}_{\boldsymbol{v}} & \mathbf{1}\oplus\boldsymbol{\xi}_{\boldsymbol{u}\oplus\boldsymbol{v}} \end{array}$$

where $\boldsymbol{\xi}_{\boldsymbol{c}}$, $\boldsymbol{\xi}_{\boldsymbol{u}}$, $\boldsymbol{\xi}_{\boldsymbol{v}}$, and $\boldsymbol{\xi}_{\boldsymbol{u}\oplus\boldsymbol{v}}$ are the truth tables of $f(\boldsymbol{c}\oplus\boldsymbol{x})$, $f(\boldsymbol{u}\oplus\boldsymbol{x})$, $f(\boldsymbol{v}\oplus\boldsymbol{x})$, and $f(\boldsymbol{u}\oplus\boldsymbol{v}\oplus\boldsymbol{x})$, respectively.

Now, it is clear that $F(\boldsymbol{y},\boldsymbol{x}) \neq G(\boldsymbol{y},\boldsymbol{x})$, because $F(\boldsymbol{y},\boldsymbol{x})$ has three blocks $\boldsymbol{\xi}$ and $G(\boldsymbol{y},\boldsymbol{x})$ only has two blocks $\boldsymbol{\xi}$.

Also, $F(\boldsymbol{y},\boldsymbol{x}) \neq H(\boldsymbol{y},\boldsymbol{x})$, because $F(\boldsymbol{y},\boldsymbol{x})$ has three blocks $\boldsymbol{\xi}$ and $H(\boldsymbol{y},\boldsymbol{x})$ only has one block $\boldsymbol{\xi}$.

Finally, $G(\boldsymbol{y},\boldsymbol{x}) \neq H(\boldsymbol{y},\boldsymbol{x})$, because $G(\boldsymbol{y},\boldsymbol{x})$ has two blocks $\boldsymbol{\xi}$ and $H(\boldsymbol{y},\boldsymbol{x})$ only has one block $\boldsymbol{\xi}$. □

Another question that appears at this point is the following: Let $F(\boldsymbol{y},\boldsymbol{x})$, $G(\boldsymbol{y},\boldsymbol{x})$, and $H(\boldsymbol{y},\boldsymbol{x})$ be bent functions of $n+2$ variables, constructed according to Corollaries 3.6 and 3.7, and Theorem 3.6 but starting with bent functions of n variables $f(\boldsymbol{x})$, $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$, respectively; if $f(\boldsymbol{x}) \neq h(\boldsymbol{x})$ and $g(\boldsymbol{x}) \neq h(\boldsymbol{x})$ is it possible that $F(\boldsymbol{y},\boldsymbol{x}) = G(\boldsymbol{y},\boldsymbol{x})$, $F(\boldsymbol{y},\boldsymbol{x}) = H(\boldsymbol{y},\boldsymbol{x})$, or $G(\boldsymbol{y},\boldsymbol{x}) = H(\boldsymbol{y},\boldsymbol{x})$? Next theorem shows that this situation can not occurs.

Theorem 3.8. *Let $f(\boldsymbol{x})$, $g(\boldsymbol{x})$, and $h(\boldsymbol{x})$ be different bent functions of n variables. Assume that (i_0,i_1,i_2,i_3), (j_0,j_1,j_2,j_3), and (k_0,k_1,k_2,k_3) are permutations of $(0,1,2,3)$. If $F(\boldsymbol{y},\boldsymbol{x})$, $G(\boldsymbol{y},\boldsymbol{x})$, and $H(\boldsymbol{y},\boldsymbol{x})$ are the bent functions constructed in Corollaries 3.4 and 3.5, and Theorem 3.6 using the bent functions $f(\boldsymbol{x})$, $g(\boldsymbol{x})$, and $h(\boldsymbol{x})$, respectively, and the permutations (i_0,i_1,i_2,i_3), (j_0,j_1,j_2,j_3), and (k_0,k_1,k_2,k_3), respectively. Then*

$$F(\boldsymbol{y},\boldsymbol{x}) \neq G(\boldsymbol{y},\boldsymbol{x}),\ F(\boldsymbol{y},\boldsymbol{x}) \neq H(\boldsymbol{y},\boldsymbol{x}) \text{ and } G(\boldsymbol{y},\boldsymbol{x}) \neq H(\boldsymbol{y},\boldsymbol{x}).$$

Proof. According to Corollaries 3.4, 3.5, and Theorem 3.6 we have that

$$F(\boldsymbol{y},\boldsymbol{x}) = f(\boldsymbol{x}) \oplus m_{i_3}(\boldsymbol{y}),$$

$$
\begin{aligned}
G(\boldsymbol{y},\boldsymbol{x}) &= (m_{j_0}(\boldsymbol{y}) \oplus m_{j_1}(\boldsymbol{y})) \oplus g(\boldsymbol{x}) m_{j_0}(\boldsymbol{y}) g(\boldsymbol{c} \oplus \boldsymbol{x}) \\
&\quad \oplus m_{j_3}(\boldsymbol{y})\,(1 \oplus g(\boldsymbol{c} \oplus \boldsymbol{x}))\,, \\
H(\boldsymbol{y},\boldsymbol{x}) &= m_{k_0}(\boldsymbol{y}) h(\boldsymbol{x}) \oplus m_{k_1}(\boldsymbol{y}) h(\boldsymbol{u} \oplus \boldsymbol{x}) \oplus m_{k_2}(\boldsymbol{y}) h(\boldsymbol{v} \oplus \boldsymbol{x}) \\
&\quad \oplus m_{k_3}(\boldsymbol{y})\,(1 \oplus h(\boldsymbol{u} \oplus \boldsymbol{v} \oplus \boldsymbol{x}))\,.
\end{aligned}
$$

If $\boldsymbol{\xi}_f$, $\boldsymbol{\xi}_g$ and $\boldsymbol{\xi}_h$ are the truth tables of $f(\boldsymbol{x})$, $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$, respectively, then the truth tables of $F(\boldsymbol{y},\boldsymbol{x})$, $G(\boldsymbol{y},\boldsymbol{x})$, and $H(\boldsymbol{y},\boldsymbol{x})$ have four blocks (not necessarily in that order and not the same order for all):

$$F: \quad \boldsymbol{\xi}_f \quad \boldsymbol{\xi}_f \quad \boldsymbol{\xi}_f \quad \boldsymbol{\xi}_f \oplus \mathbf{1} \tag{6}$$

$$G: \quad \boldsymbol{\xi}_g \quad \boldsymbol{\xi}_g \quad \boldsymbol{\xi}_{g,\boldsymbol{c}} \quad \mathbf{1} \oplus \boldsymbol{\xi}_{g,\boldsymbol{c}} \tag{7}$$

$$H: \quad \boldsymbol{\xi}_h \quad \boldsymbol{\xi}_{h,\boldsymbol{u}} \quad \boldsymbol{\xi}_{h,\boldsymbol{v}} \quad \mathbf{1} \oplus \boldsymbol{\xi}_{h,\boldsymbol{u}\oplus\boldsymbol{v}} \tag{8}$$

where $\boldsymbol{\xi}_{g,\boldsymbol{c}}$, is the truth table of $g(\boldsymbol{c} \oplus \boldsymbol{x})$; $\boldsymbol{\xi}_{h,\boldsymbol{u}}$, $\boldsymbol{\xi}_{h,\boldsymbol{v}}$, and $\boldsymbol{\xi}_{h,\boldsymbol{u}\oplus\boldsymbol{v}}$ are the truth tables of $h(\boldsymbol{u} \oplus \boldsymbol{x})$, $h(\boldsymbol{v} \oplus \boldsymbol{x})$, and $h(\boldsymbol{u} \oplus \boldsymbol{v} \oplus \boldsymbol{x})$, respectively.

Since $f(\boldsymbol{x}) \neq g(\boldsymbol{x})$, we have that

$$f(\boldsymbol{x}) = m_i(\boldsymbol{x}) \oplus f'(\boldsymbol{x}) \quad \text{and} \quad g(\boldsymbol{x}) = m_j(\boldsymbol{x}) \oplus g'(\boldsymbol{x})$$

with $i \neq j$, $m_i(\boldsymbol{x})$ is a minterm that is not in the expression of $g'(\boldsymbol{x})$ as a sum of minterms, and $m_j(\boldsymbol{x})$ is a minterm that is not in the expression of $f'(\boldsymbol{x})$ as a sum of minterms.

So, according to expression (6), in the ith position of each one of the four blocks of the truth table of $F(\boldsymbol{y},\boldsymbol{x})$ we have

$$1 \quad 1 \quad 1 \quad 0$$

but, according to expression (7), in the ith position of each one of the four blocks of the truth table of $G(\boldsymbol{y},\boldsymbol{x})$ we have

$$0 \quad 0 \quad ? \quad ?$$

depending on $\boldsymbol{c}$. If $\boldsymbol{e}_j \oplus \boldsymbol{c} = \boldsymbol{e}_i$ we have

$$0 \quad 0 \quad 1 \quad 0$$

but for $\boldsymbol{e}_j \oplus \boldsymbol{c} \neq \boldsymbol{e}_i$, then we have

$$0 \quad 0 \quad 0 \quad 1$$

In any case, it is clear that both truth tables are different. So, $F(\boldsymbol{y}, \boldsymbol{x}) \neq G(\boldsymbol{y}, \boldsymbol{x})$.

The same applies to the rest of the cases. □

References

[1] A. Canteaut and P. Charpin. Decomposing bent functions. *IEEE Transactions on Information Theory*, 49(8): 2004–2019 (2003).

[2] C. Carlet and P. Guillot. A characterization of binary bent functions. *Journal of Combinatorial Theory (Series A)*, 76: 328–335 (1996).

[3] D. K. Chang. Binary bent sequences of order 64. *Utilitas Mathematica*, 52: 141–151 (1997).

[4] C. Charnes, M. Rötteler and T. Beth. Homogeneous bent functions, invariants, and designs. *Designs, Codes and Cryptography*, 26: 139–154 (2002).

[5] J. F. Dillon. *Elementary Hadamard Difference Sets*. PhD Thesis, University of Maryland, 1974.

[6] X.-D. Hou. $GL(m, 2)$ acting on $R(r, m)/R(r-1, m)$. *Discrete Mathematics*, 149: 99–122 (1996).

[7] X.-D. Hou. Cubic bent functions. *Discrete Mathematics*, 189: 149–161 (1998).

[8] X.-D. Hou. On the coefficients of binary bent functions. *Proceedings of the American Mathematical Society*, 128(4): 987–996 (1999).

[9] X.-D. Hou and P. Langevin. Results on bent functions. *Journal of Combinatorial Theory (Series A)*, 80: 232–246 (1997).

[10] R. McFarland. A family of noncyclic difference sets. *Journal of Combinatorial Theory (Series A)*, 15: 1–10 (1973).

[11] C. Qu, J. Seberry and J. Pieprzyk. On the symmetric property of homogeneous Boolean functions. In J. Pieprzyk, R. Safavi-Naini and J. Seberry (editors), *Information Security and Privacy – ACISP'99*, volume 1587 of *Lecture Notes in Computer Science*, pages 26–35. Springer-Verlag, Berlin, 1999.

[12] O. S. Rothaus. On "bent" functions. *Journal of Combinatorial Theory (Series A)*, 20: 300–305 (1976).

[13] J. Seberry, X.-M. Zhang and Y. Zheng. Nonlinearly balanced Boolean functions and their propagation characteristics (extended abstract). In D. R. Stinson (editor), *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 40–60. Springer-Verlag, Berlin, 1994.

[14] J. Seberry, X.-M. Zhang and Y. Zheng. Nonlinearity and propagation characteristics of balanced Boolean functions. *Information and Computation*, 119: 1–13 (1995).

Boolean Functions: Cryptography and Applications BFCA'07
Fonctions Booléennes : Cryptographie & Applications

SOME NECESSARY CONDITIONS FOR A QUADRATIC FEEDBACK SHIFT REGISTER TO GENERATE A MAXIMUM LENGTH SEQUENCE

Ali Doğanaksoy[1, 2], Elif Saygı[2, 3] and Zülfükar Saygı[2]

Abstract. In this paper, we deal with the properties of the quadratic feedback shift registers generating maximum length sequences. We give some necessary conditions for a quadratic feedback function f of a feedback shift register to generate a maximum length sequence. Also we present a method generalizing this condition. Instead of searching all the sequence, looking at the algebraic normal form of the function f we can understand if the corresponding shift register generates a sequence having short period.
Keywords: Boolean functions, binary sequences, maximum length sequences, m-sequences, shift registers, stream ciphers.

1. Introduction

In many fields, such as mathematics, computer science, communication, cryptography, networks etc., binary sequences are used extensively [1–3]. A common way of producing such a sequence is employing a feedback shift register (FSR). It turns out to be an important problem to examine the properties of sequences produced

[1] Department of Mathematics, Middle East Technical University
İnönü Bulvarı, 06531, Ankara, Turkey
[2] Institute of Applied Mathematics, Middle East Technical University
İnönü Bulvarı, 06531, Ankara, Turkey
[3] Faculty of Education, Hacettepe University, Ankara, Turkey,
email: {aldoks,e110925,saygi}@metu.edu.tr
Phone: +90 (312) 210 53 54 Fax: +90 (312) 210 29 85

J-F. Michon, P. Valarcher, J-B. Yunès (Eds.): BFCA'07

by FSRs and as a consequence, FSRs have been widely studied in the literature [4, 5].

A FSR of length n consists of a pair (F, f) where f is a Boolean function defined on n variables and F is a vectorial Boolean function defined on n variables by setting

$$F(x_1, x_2, \ldots, x_n) = (x_2, \ldots, x_n, f(x_1, x_2, \ldots, x_n)),$$

$x_1, \ldots, x_n \in F_2$. F, in fact, describes how to obtain the new state from the previous state and thus called the next-state function; f is used to define the nth term of the new state and is called the feedback function. Starting from an initial state $(s_1, s_2, \ldots, s_n)$, we obtain the sequence of states:

$$F(s_1, s_2, \ldots, s_n) = (s_2, \ldots, s_n, s_{n+1}),$$

$$F(s_2, \ldots, s_{n+1}) = (s_3, \ldots, s_{n+1}, s_{n+2}),$$

$$\vdots$$

Then, by definition, the binary sequence produced by the FSR is $s_1, s_2, \ldots, s_n, s_{n+1}, \ldots$.

A FSR is a finite state machine (with n stages), thus it has a finite number (at most 2^n) of states and it produces an ultimately periodic sequence (of period at most 2^n). In many situations a FSR is not allowed to reach the all 0 state and in such a case the output sequence has period at most $2^n - 1$. One of the interesting problems is the analysis of the possible periods of FSRs. The most interesting sequences are the ones with maximum periods. A binary sequence of span n is called a maximal length sequence (m-sequence) if its period (or length) is $2^n - 1$.

If the feedback function f of a FSR is a linear function we call the FSR a linear feedback shift register (LFSR). Maximal length sequences generated by linear LFSR's have longest period and good statistical properties. On the other hand, there exist several algorithms to calculate the span of a given LFSR, among which the best known is the Berlecamp-Massey algorithm [1,6]. By using this algorithm a binary m-sequence of span n can be completely determined by observing any consequtive $2n$ bits of the sequence. This, of course, means a cryptographical weakness. In the design of stream ciphers, when the LFSRs are used, to avoid this weakness an amount of nonlinearity has been added by means of a nonlinear filter function or a nonlinear combiner [4, 5]. Another possibility

is to use FSRs having nonlinear feedback functions. In this work we deal with Quadratic Feedback Shift Registers (QFSRs), that is whose feedback functions are quadratic Boolean Functions.

The feedback function of a QFSR can be represented as follows

$$f(x_0, x_1, \ldots, x_n) = \bigoplus_{0 \le i \le n} a_i x_i \oplus \bigoplus_{0 \le i < j \le n} a_{i,j} x_i x_j.$$

The aim of this paper is to analyze the properties of the quadratic feedback shift registers which generate maximum length sequences. We obtain some necessary conditions for the feedback function f to generate a maximum length sequence. Also we present a method to generalize the main idea. Instead of examining the sequence, one can concantrate on algebraic normal form of the Boolean function f to understand whether the corresponding shift register generates a sequence having a short period.

2. Some Previous Results

In this section we give some previous results and some necessary conditions to generate quadratic m-sequences.

For a FSR if some initial state is never repeated, the generated cycle of states is said to have a branch point, that is, for some different states S_1 and S_2, we have $F(S_1) = F(S_2)$. We now state some results whose proofs can be found in the mentioned references.

Proposition 2.1 ([7]). *A sequence have no branch point if and only if the next state function F is one to one.*

Recall that the degree of a Boolean function f, denoted by $deg(f)$, is the maximum number of variables in a term appearing in f. The degree of a variable x_i in f, denoted by $deg_f(x_i)$, is the maximum number of variables among all terms in which x_i appears.

Proposition 2.2 ([7]). *A next state function F is one to one if and only if $deg_f(x_0) = 1$.*

Note that, $deg_f(x_0) = 1$ means that f is of the form $f = x_0 + g$ where g is a Boolean function which does not depend on x_0.

Corollary 2.3. *Let f be a feedback function of a FSR that generates a quadratic m-sequence. Then $deg_f(x_0) = 1$.*

3. Main Results

In this section we present our main results. Our results give conditions for a feedback function f of a FSR to generate a maximum length sequence. Throughout this section f is a quadratic Boolean function of the form

$$\begin{aligned} f(x_0, x_1, \ldots, x_n) &= x_0 \oplus g(x_1, \ldots, x_n) \\ &= x_0 \oplus \bigoplus_{1 \le i \le n} a_i x_i \oplus \bigoplus_{1 \le i < j \le n} a_{i,j} x_i x_j. \quad (1) \end{aligned}$$

Now we state a useful proposition which is stated and proved in [8].

Proposition 3.1. *Let $f = x_0 + g$ be a quadratic feedback function which generates an m-sequence. Then,*

a. The number of linear terms in g and the number of quadratic terms in g are not equal in modulo 2,
b. There is at least one linear term in g.

The main idea in the proof of the above proposition is considering certain periodical structures of the given sequence. The motivation behind our work is to generalize this idea to a larger class of templates.

We now examine the cases where the sequence has a cycle σ of length m. Here we first present some useful notations to have used in the remaining part of the paper. Let f be an $n+1$ variable Boolean function of the form $f(x_0, x_1, \ldots, x_n) = x_0 \oplus g(x_1, \ldots, x_n)$. For $i = 1, 2, \ldots m$, $t = \lfloor (n-i)/m \rfloor$ and for any $r \le m$ we define

$$\begin{aligned} y_i^m &= \bigoplus_{0 \le k \le t} a_{i+mk} \oplus \bigoplus_{0 \le k < l \le t} a_{i+mk, j+ml}, \\ y_i^m * y_j^m &= \bigoplus_{0 \le k \le t} a_{i+mk, j+mk}, \\ y_{i_1, i_2}^m &= y_{i_1}^m \oplus y_{i_2}^m \oplus y_{i_1}^m * y_{i_2}^m, \\ y_{i_1, i_2, \cdots, i_r}^m &= \bigoplus_{1 \le k \le r} y_{i_k}^m \oplus \bigoplus_{1 \le k < l \le r} y_{i_k}^m * y_{i_l}^m. \end{aligned}$$

Note that the above defined $y_{i_1, i_2, \cdots, i_r}^m$'s are the x-or of some specific coefficients of the function defined in (1). Before introducing the main result, for the sake of some clarification, we present an example.

Example 3.2. Let σ be a cycle of length $m = 4$, and define f to be a function with 10 variables $x_0, x_1, \ldots, x_9$. Now we calculate y_1^4, y_2^4, $y_1^4 * y_2^4$ and $y_{1,2}^4$.

$$\begin{aligned} y_1^4 &= a_1 \oplus a_5 \oplus a_9 \oplus a_{1,5} \oplus a_{1,9} \oplus a_{5,9}, \\ y_2^4 &= a_2 \oplus a_6 \oplus a_{2,6}, \\ y_1^4 * y_2^4 &= a_{1,2} \oplus a_{1,6} \oplus a_{2,5} \oplus a_{2,9} \oplus a_{5,6} \oplus a_{6,9}, \\ y_{1,2}^4 &= y_1^4 \oplus y_2^4 \oplus y_1^4 * y_2^4. \end{aligned}$$

The following propositions state new necessary conditions for the sequence generated by QFSRs to be a maximum length sequence.

Proposition 3.3. *Let f be an $n+1$ variable quadratic feedback function which generates an m-sequence then,*

a. $y_1^2 \oplus y_2^2 \oplus y_1^2 \cdot y_2^2 = 1$, *if* $n + 1 \equiv 0 \ mod\ 2$,
b. $y_1^2 \cdot y_2^2 = 0$, *if* $n + 1 \equiv 1 \ mod\ 2$.

Proof. The proofs of these statements are similar. The main idea behind is that a feedback function which generates an m-sequence can not generate the cycle (01).

Assume that $n+1 \equiv 0 \bmod 2$ and the function f generates the (01) cycle. That is,

$$f(0101\cdots01) = 0 \text{ if and only if } g(1010\cdots01) = 0 \text{ and}$$

$$f(1010\cdots010) = 1 \text{ if and only if } g(0101\cdots010) = 0.$$

By definition, we have $y_1^2 = 0$ and $y_2^2 = 0$. Since f can not generate the (01) cycle, both of y_1^2 and y_2^2 can not be zero, that is to say $y_1^2 \oplus y_2^2 \oplus y_1^2 \cdot y_2^2 = 1$.

Similarly if $n+1 \equiv 1 \bmod 2$ and f generates the (01) cycle, we have $f(0101\cdots010) = 1$, $f(1010\cdots0101) = 0$ that is equivalent to $g(1010\cdots010) = 1$, $g(0101\cdots0101) = 1$. So we have $y_1^2 = 1$ and $y_2^2 = 1$. But we know that f can not generate the (01) cycle, so both of y_1^2 and y_2^2 can not be one, which is equivalent to say $y_1^2 \cdot y_2^2 = 0$. □

Example 3.4. Let f_1, f_2, f_3, f_4 be quadratic Boolean functions of the form (1) and defined as $f_1 = x_0+x_3+x_1x_{16}+x_{18}+x_{19}+x_{14}x_{19}$, $f_2 = x_0 + x_6 + x_1x_3 + x_5x_7$, $f_3 = x_0 + x_1 + x_2 + x_3 + x_4 +$

$x_2x_9 + x_5x_{11} + x_6x_9$ and $f_4 = x_0 + x_1 + x_3 + x_1x_3 + x_2x_4 + x_3x_4$. Then by checking the conditions given in Proposition 3.3 we see that the functions f_1, f_2 and f_3 satisfy the conditions, that is the QFSRs with the corresponding feedback functions f_1, f_2 and f_3 may generate m-sequences. But for f_4 we have $n + 1 = 5 \equiv 1 \mod 2$, $y_1^2 = a_1 + a_3 + a_{1,3} = 1$ and $y_2^2 = a_2 + a_4 + a_{2,4} = 1$. Therefore the QFSR having feedback function f_4 cannot generate an m-sequence with the Proposition 3.3.

Proposition 3.5. *Let f be an $n + 1$ variable quadratic feedback function which generates an m-sequence then,*

a. $y_1^3 \oplus y_2^3 \oplus y_3^3 \oplus y_1^3 \cdot y_2^3 \oplus y_1^3 \cdot y_3^3 \oplus y_2^3 \cdot y_3^3 = 1$, *if* $n + 1 \equiv 0$ *mod* 3,
b. $y_1^3 \cdot y_2^3 \cdot y_3^3 \oplus y_1^3 \cdot y_3^3 = 0$, *if* $n + 1 \equiv 1$ *mod* 3
c. $y_1^3 \cdot y_2^3 \cdot y_3^3 \oplus y_2^3 \cdot y_3^3 = 0$, *if* $n + 1 \equiv 2$ *mod* 3
d. $y_{2,3}^3 \oplus y_{1,3}^3 \oplus y_{1,2}^3 \oplus y_{2,3}^3 \cdot y_{1,3}^3 \oplus y_{2,3}^3 \cdot y_{1,2}^3 \oplus y_{1,3}^3 \cdot y_{1,2}^3 = 1$, *if* $n + 1 \equiv 0$ *mod* 3
e. $y_{2,3}^3 \cdot y_{1,3}^3 \cdot y_{1,2}^3 \oplus y_{2,3}^3 \cdot y_{1,2}^3 = 0$, *if* $n + 1 \equiv 1$ *mod* 3
f. $y_{2,3}^3 \cdot y_{1,3}^3 \cdot y_{1,2}^3 \oplus y_{1,3}^3 \cdot y_{1,2}^3 = 0$, *if* $n + 1 \equiv 2$ *mod* 3

Proof. To prove the statements we use the same technique as in the proof of Proposition 3.3. Instead of using the (01) cycle we concentrate on the cycles (001) and (011). Using the definitions of $y_{i_1,i_2,\cdots,i_r}^m$'s and taking the feedback function f as in (1) we complete the proof. □

Example 3.6. Let f_1, f_2, f_3 be quadratic Boolean functions defined as in the Example 3.4. We know that the condition given in Proposition 3.3 is satisfied by f_1, f_2 and f_3. An easy computation shows that the conditions given in Proposition 3.5 are satisfied for the functions f_1 and f_2. But for f_3 we have $n + 1 = 12 \equiv 0 \mod 3$ and $y_1^3 = y_2^3 = y_3^3 = 0$. Therefore the QFSR with the feedback function f_3 cannot generate an m-sequence, with the Proposition 3.5.

Remark 3.7. *Proposition 3.3 and Proposition 3.5 give necessary conditions for a QFSR to generate an m-sequence. By examining the algebraic normal form of a quadratic function one can easily check the conditions given in the propositions to see whether the corresponding QFSR is able to generate an m-sequence or not.*

The idea in the proof of Proposition 3.3 and Proposition 3.5 can be easily generalized to cycles of length greater than three.

Theorem 3.8. *Let f be an $n+1$ variable quadratic feedback function generating the cycle σ of length m and Hamming weight w. If $n+1 \equiv 0 \ (mod\ m)$ then $y_i^m = 0$, $y_{i_1,i_2}^m = 0$, ..., $y_{j_1,j_2,\cdots,j_w}^m = 0$, where $1 \leq i \leq m$, $1 \leq i_1 < i_2 \leq m$ and $1 \leq j_1 < j_2 < \cdots < j_w \leq m$.*
If $n+1 \equiv t \ (mod\ m)$ then $y_t^m = 1$, $y_{i_1,t}^m = y_{t,i_2}^m = 1$, ..., $y_{j_1,j_2,\cdots,j_w}^m = 1$, where $1 \leq i_1 < t \leq m-1$, $1 \leq t < i_2 \leq m-1$ and $1 \leq j_1 < j_2 < \cdots < j_w \leq m-1$ with $j_k = t$ for some k. Also $y_m^m = 1$, $y_{i_1,m}^m = 1$, ..., $y_{j_1,j_2,\cdots,m}^m = 1$, where $i_1 \neq t$, $1 \leq i_1 \leq m-1$, $1 \leq j_1 < j_2 < \cdots \leq m-1$ and $j_k \neq t$ for any k.

Proof. Similar to that of Proposition 3.3 and Proposition 3.5 □

References

[1] J. L. Massey, Shift register synthesis and BCH decoding, *IEEE Trans. Inf. Theory*, vol. IT-15, no. 1, pp. 122127, Jan. (1969).

[2] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*, New York: Wiley, (1996).

[3] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt, *Spread Spectrum Communications Handbook*, New York: McGraw-Hill, (1994).

[4] S. W. Golomb, *Shift register sequences*, Aegean Park Press, Laguna Hills, California, (1982).

[5] R. A. Rueppel, *Analysis and design of stream ciphers*, Springer Verlag, Berlin, (1986).

[6] E. R. Berlekamp, *Algebraic Coding Theory*, New York: McGraw-Hill, (1968).

[7] A. Doğanaksoy and E. Saygı, On The Quadratic Feedback Shift Registers, *I. Ulusal Kriptoloji Sempozyumu Bildiriler Kitabı*, pp. 127-133, (2005). http://www.iam.metu.edu.tr/sempozyum/2005/

[8] A. H. Chan, R. A. Games and J. J. Rushanan, On the quadratic m-sequences, *Proceedings of Fast Software Encryption*, LNCS 809, pp. 166-173, (1994).

Boolean Functions: Cryptography and Applications BFCA'07
Fonctions Booléennes : Cryptographie & Applications

IMPROVED ALGORITHM TO FIND EQUATIONS FOR ALGEBRAIC ATTACKS FOR COMBINERS WITH MEMORY

Frederik Armknecht[1], Pierre-Louis Cayrel[2], Philippe Gaborit[2] and Olivier Ruatta[2]

Abstract. Algebraic attacks have established as an important tool for cryptanalyzing LFSR-based keystream generators. Crucial for an efficient attack is to find appropriate equations of a degree as low as possible. Hereby, lower degrees are possible if many keystream bits are involved in one equation. An example is the keystream generator E_0 employed in Bluetooth, where equations of degree 4 exist for $r = 4$ and 5 clocks but equations of degree 3 for $r \approx 8,822,188$. The existence of degree 3 equations with $5 < r \ll 8,822,188$ clocks remained an open question. It is known that valid equations correspond to annihilators of certain sets. The effort to compute the sets and to find annihilators on them are exponential in r, making efficient algorithms desirable. We describe first in this paper several improvements for computing the sets and their annihilators (the intersection method). Second, we use our new improvements to exclude the existence of degree 3 equations for E_0 with $5 < r \leq 9$. We also find 4 degree 4 annihilators with 7 consecutive output bits.
Keywords: Algebraic attacks, low degree equations, annihilators, algorithm

[1] NEC Europe Ltd. Network Laboratories, Kurfrsten-Anlage 36, D-69115 Heidelberg. email: Frederik.Armknecht@netlab.nec.de

[2] Université de Limoges, XLIM-DMI, 123, Av. Albert Thomas, 87000 Limoges, France.
email: {pierre-louis.cayrel,philippe.gaborit,olivier.ruatta}@xlim.fr

J-F. Michon, P. Valarcher, J-B. Yunès (Eds.): BFCA'07

1. Introduction

Shortly speaking, an algebraic attacks consists in generating and solving a system of (non-)linear equations which describe implicitly the secret key in dependence on known values. In the case of keystream generators, this involves the known keystream bits. Stream ciphers are designed for online encryption of secret plaintext bitstreams which have to pass an insecure channel. Widely used in practice are stream ciphers are keystream generators based on linear feedback shift registers (LFSR), a prominent example being the E_0 keystream generator used in the Bluetooth standard for wireless communications [4]. Before exchanging data, the keystream generator is initialized with some secret value $\mathcal{S}_0 \in \{0,1\}^n$ on which both sender and receiver have to agree on. To encrypt a stream of plaintext bits $p_0, p_1, \ldots$, the keystream generator is used to generate a bitstream $z_0, z_1, \ldots$ of the same length. Both streams are XORed bit by bit, giving the cipher text bits $c_t := p_t \oplus z_t$. The bits c_t are send to the receiver, who knows the secret initialization value of the keystream generator and therefore can produce the same keystream bits z_t to decrypt the message by $p_t = c_t \oplus z_t$.

To evaluate the security, it is assumed that a potential adversary knows the specifications of the keystream generator and some of the keystream bits z_t. An attack is to recover the value of $\mathcal{S}_0$ using the given information. Over the last years, several kind of attacks have been invented (e.g., fast correlation attacks [6,14,15,21], backtracking attacks [12, 13, 24, 25], time-memory tradeoffs [5], BDD-based attacks [16] etc.). All have in common that the attack complexity is exponential in the key size n.

Recently, a new kind of attack was proposed: algebraic attack. For some ciphers, algebraic attacks outmatched all previously known attacks (e.g. [1, 9]). The basic idea is to generate a system of equations with its solution being the secret key. Although solving system of non-linear equations over finite fields is a NP-hard problem, in this case the equations have some particular properties. Each equation provides some information on the secret key depending on r successive known keystream bits $z_t, \ldots, z_{t+r-1}$ and the degree is upper bounded by some value d which is independent of n. If the number of linearly independent equations is

high enough, an attack can be performed within $O(n^{3d})$ operations. Observe that, despite to the other attacks, the complexity is only polynomial in n but exponential in d.

Thus, one of the major topics in the context of algebraic attacks is the search for low degree equations. Whereas the general existence of such equations has been proven in [1], only few is known on how to find (or avoid) equations of a degree d. For example, in the case of E_0, an equation of degree 4 over $r = 4$ clocks has been developed in [1], what led to fastest attack at this time. Later, in [7], these equations were combined to derive a new equation of degree 3 over $r \approx 8,882,198$ clocks, what reduced the time effort by several magnitudes. As far as we know, algebraic attacks are still the fastest attacks on stand-alone E_0.[1] Furthermore, these results show the possibility that for $r > 4$, equations with an degree < 3 may exist. This is particulary interesting as equations of degree 2 or 1 would lead to practical attacks. Whereas it has been confirmed that no equations of degree < 4 exist for E_0 for $r \leq 5$, to the best of our knowledge, no results exist for $r \geq 6$.

So far, all methods published to find all (or exclude the existence of) degree d equations are based on computing annihilators of certain sets (see [1, 20] and for a general framework [2]). However, as the generation of these sets and thus the effort for computing the annihilators grow exponentially with r, efficient algorithms for both steps are crucial. Normally, finding annihilators is done by Gaussian elimination. Recently, an improved algorithm was proposed in [3] which reduced the complexity of the Gaussian elimination in these cases from cubic to quadratic, allowing the computation of the algebraic immunity for Boolean functions with more variables than it was possible before. Unfortunately, it is difficult to achieve this complexity in practice and requires a sophisticated memory management.

Our results are the following. First, we describe how to compute a basis of all low degree equations over r clocks and present several improvements on the way to construct the set X_Z introduced in [2] and on the way to prove the non existence of certain low degree annihilators. Then, we give some experimental results on the non existence of low degree equations of degree three for E_0 for a number of cloks between 5 and 9.

[1]Faster attacks on Bluetooth have been described in [17–19] but they rely on the linearity of the key schedule of the Bluetooth encryption system.

The paper is structured as follows. After explaining algebraic attacks and the connection betweem equations and annihilators in Section 2, we describe how to compute the corresponding sets and some improvements in Sections 3 and 4. Depending on a given secret information $x^* \in \{0,1\}^n$, the stream cipher produces a keystream $Z(x^*) = (z_1, z_2, \cdots)$ which is bitwise XORed with E. Knowing x^*, the decryption can be performed by using the same rule. It is common to evaluate the security of a stream cipher relative to the pessimistic scenario that an attacker has access not only to the encrypted bitstream, but even to a sufficiently long piece of keystream. Thus, the cryptanalysis problem of a given stream cipher consists in computing the secret information x^* from a sufficiently long prefix of $Z(x^*)$.

We call a stream cipher LFSR-based, if it consists of a certain number k of linear feedback shift registers (LFSRs) and an additional device, called the nonlinear combiner, which transforms the internal linear bitstream, produced by the LFSRs, into a nonlinear output keystream. Because of the simplicity of LFSRs and the excellent statistical properties of bitstreams produced by well-chosen LFSRs, LFSR-based stream ciphers are widely used in practice. A lot of different nontrivial approaches to the cryptanalysis of LFSR-based stream ciphers (fast correlation attacks, backtracking attacks, time-space tradeoffs, BDD-based attacks etc.) were discussed in the relevant literature, and a lot of corresponding design criterions (correlation immunity, large period and linear complexity, good local statistics etc.) for such stream ciphers were developed.

A (k,l)-combiner consists of k LFSRs and a finite Mealy automaton with k input bits, one output bit and l memory bits. Let n be the sum of the lengths of the k LFSRs. Starting from a secret initial assignment $x^* \in \{0,1\}^n$, the LFSRs produce an internal linear bitstream $L(x^*)$, built by blocks x^t of k parallel bits for each clock t. Starting from a secret initial assignment $c^1 \in \{0,1\}^l$ to the memory bits, in each clock t the automaton produces the t-th keystream bit z_t corresponding to x^t and c^t and changes the inner state to c^{t+1} (see figure 1). The secret information is given by x^* and c^1. Numerous ciphers of this type are used in practice. Note, e.g., that the E_0 keystream generator used in the Bluetooth wireless LAN system (see Bluetooth SIG (2001) [4]) is a $(4,4)$-combiner.

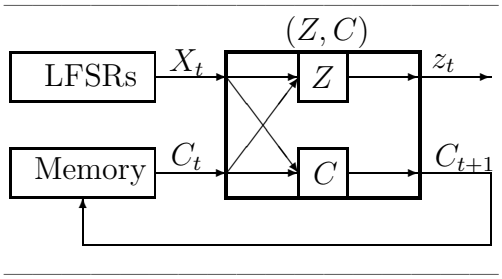


FIGURE 1. A (k, l)-combiner

The aim of this paper is to analyze the security of (k, l)-combiners with respect to algebraic attacks, a new method for attacking stream and block ciphers. Algebraic attacks exist against AES and Serpent (Courtois and Pieprzyk (2002) [8]) and Toyocrypt (Courtois (2003) [9]). Related algebraic attacks were used to attack the HFE public key cryptosystem (Courtois (2002) [8]).

Courtois and Meier (2003) discussed algebraic attacks on general LFSR-based stream ciphers and presented the best known attacks on Toyocrypt and LILI-128 so far [9]. Very recently, Courtois introduced fast algebraic attacks on LFSR-based stream ciphers, an improved version of the algebraic attacks (Courtois (2003) [7]).

An algebraic attack is based on a nontrivial low degree relation p for r clocks, i.e. a relation which holds for any sequence of r consecutive bits of the keystream and the corresponding kr internal bits. Given such a relation p of small degree d and a sufficiently long piece of a keystream $Z(x^*, c^1)$, p can be used to produce an overdefined system of T nonlinear equations in the initial bits of the LFSRs, which can be thought of as system of linear equations in the monomials of length at most d. If T is large enough then we get a unique solution which is induced by x^*, and from which x^* can be derived in a straightforward way.

2. Algebraic attacks and annihilators

In this section, we describe basic ideas of algebraic attacks on keystream generators and the connection between valid equations and annihilators of given sets. Hereby, an annihilator of a set $X \subseteq \mathbb{F}_2^n$ with $\mathbb{F}_2$ being the finite field with two elements is defined

to be a function $g : \mathbb{F}_2^n \to \mathbb{F}_2$ with $g(x) = 0$ for all $x \in X$. First of all, we define (k, ℓ)-combiners which is the class of keystream generators we are dealing with:

Definition 2.1. A (k, ℓ)-combiner with $k \geq 1$ and $\ell \geq 0$ consists of an internal state $\mathcal{S} \in \mathbb{F}_2^\ell \times \mathbb{F}_2^n$, a regular matrix L over $\mathbb{F}_2$ of size $n \times n$, called the LFSR feedback matrix, a (projection) matrix P over $\mathbb{F}_2$ of size $n \times k$ a non-linear next memory state function $\Psi : \mathbb{F}_2^\ell \times \mathbb{F}_2^k \to \mathbb{F}_2^\ell$ and an output function $f : \mathbb{F}_2^\ell \times \mathbb{F}_2^k \to \mathbb{F}_2$.

One example for a practically used (k, ℓ)-combiner is the E_0 keystream generator from the Bluetooth standard, which is a $(4, 4)$-combiner with $n = 128$.

The generation of the keystream works as follows. First, the internal state $\mathcal{S}$ is initialized to $\mathcal{S}_0 := (Q_0, K) \in \mathbb{F}_2^\ell \times \mathbb{F}_2^n$ where $K \in \mathbb{F}_2^n$ is the LFSRs initial state and the register of $Q_0 \in \mathbb{F}_2^\ell$ is called the memory. The content of the LFSRs is updated by a linear function, here denoted by the matrix L. In many cases, only some of the LFSRs' outputs are used for the computation of the actual keystream bit and the next state of the memory register. This is expressed by the projection matrix P and means that only the values in $K_t := K \cdot L^t \cdot P$ are involved in the computations at clock t. The memory is updated via $Q_{t+1} := \Psi(Q_t, K_t)$, whereas the keystream bit z_t is computed by $z_t = f(Q_t, K_t)$.

Obviously, the values of the keystream bits $z_t, \ldots, z_{t+i}$ depend only on Q_t and $K_t, \ldots, K_{t+i}$. We express this fact by defining the extended output function $f^r(Q_t, K_t, \ldots, K_{t+r-1}) = (z_t, \ldots, z_{t+i})$. More formally, it holds that

$$\begin{aligned} f^r(Q_t, K_t, \ldots, K_{t+i}) &= (f(Q_t, K_t), f(Q_{t+1}, K_{t+1}), \ldots, \\ &\quad f(Q_{t+r-1}, K_{t+r-1})) \end{aligned}$$

with $Q_{i+1} = \Psi(Q_i, K_i)$. An algebraic attack is based on finding functions $F_1, \ldots, F_s : \mathbb{F}_2^{k \cdot r} \times \mathbb{F}_2^r \to \mathbb{F}_2$ such that

$$F_i(K_t, \ldots, K_{t+r-1}, z_t, \ldots, z_{t+r-1}) = 0 \tag{1}$$

is true for all clocks $t \geq 0$ and $1 \leq i \leq s$. Given such functions, a system of equation can be generated by

$$F(K_{t_i}, \ldots, K_{t_i+r-1}, z_{t_i}, \ldots, z_{t_i+r-1}) = 0$$

where $z_{t_i}, \ldots, z_{t_i+r-1}$ denote r successive *known* keystream bits. If enough equations are given, the LFSRs initial state K is uniquely determined and can be found by solving the system of equation. Once K is known, Q_0 can usually be reconstructed quite easily or found by exhaustive search. Thus, we refer to K as the secret key and call n the key size.

To compute the solution, the so-called linearization method is possible. Given that each of the equations has a degree $\leq d$ with $d = \max_i\{\deg_{K_t,\ldots,K_{t+r-1}}(F_i)\}$ being independent of n, the number of involved monomials is upper bounded by $\mu(n,d) := \binom{n}{0} + \ldots + \binom{n}{d}$. If the number of linearly independent equations equals the number of occurring monomials, a solution can be computed by Gaussian elimination where each monomial is treated like an independent variable. As $\mu(n,d) \in O(n^d)$, the overall complexity is in $O(n^{3d})$, that is polynomial in the key size n but exponential in d.

Thus, it is preferable to compose the system of equations with linearly independent equations of a degree as low as possible. In particular, for known $z_t^{t+r-1} := (z_t, \ldots, z_{t+r-1})$ and corresponding unknown $K_t^{t+r-1} := (K_t, \ldots, K_{t+r-1})$, one would like avoid the case that $F_1(K_t^{t+r-1}, z_t^{t+r-1}), \ldots, F_s(K_t^{t+r-1}, z_t^{t+r-1})$ are linearly dependent and to exclude those expressions $F_i(K_t^{t+r-1}, z_t^{t+r-1})$ which have not the minimal degree. Therefore, to be sure that one uses only linearly independent equations of the minimal degree for any choice of z_t^{t+r-1}, it is advisable to derive equations for each possible value of z_t^{t+r-1} *independently*. This leads to the notion of Z-functions:

Definition 2.2. Fix a (k,ℓ)-combiner and $Z \in \mathbb{F}_2^r$. A function $F : \mathbb{F}_2^{r\cdot k} \to \mathbb{F}_2$ is called a Z-function if whenever a part $z_t^{t+r-1} := (z_t, \ldots, z_{t+r-1})$ of the keystream is equal to Z, then

$$F(K_t, \ldots, K_{t+r-1}) = 0 \text{ is valid}$$

. More formally, it must hold:

$$\forall t : z_t^{t+r-1} = Z \Rightarrow F(K_t, \ldots, K_{t+r-1}) = 0.$$

If an adversary knows (for a fixed integer r), Z-functions $F_{Z,1}, \ldots, F_{Z,s_Z}$ for each $Z \in \mathbb{F}_2^r$, then we can set up the following

system of equations:

$$0 = F_{(z_{t_i}^{t_i+r-1},j)}(K_{t_i}, \ldots, K_{t_i+r-1}), \quad 1 \leq j \leq s_Z$$

where $z_{t_i}^{t_i+r-1}$ denote as usual a sequence of r known keystream bits starting from clock t_i. More on Z-functions and a concrete example can be found in the appendix, Section A.

Of course, this arises the question on how to find a basis of Z-functions of minimal degree for any $Z \in \mathbb{F}_2^r$ for a given (k, ℓ)-combiner and a fixed value of r. For this purpose, the following theorem comes in handy. The proof is rather easy and can be found for example in [2].

Theorem 2.3. *Let a (k, ℓ)-combiner be given and f^r its extended output function. Then, a function $F : \mathbb{F}_2^{r \cdot k} \to \mathbb{F}_2$ is a Z-function with $Z \in \mathbb{F}_2^r$ if and only if F is an annihilator of the set*

$$X_Z := \{(X_1, \ldots, X_r) \in \mathbb{F}_2^{r \cdot k} \mid \exists Q \in \mathbb{F}_2^\ell : f^r(Q, X_1, \ldots, X_r) = Z\}. \tag{2}$$

If an adversary observes the values $z_t^{t+r-1} = (z_t, \ldots, z_{t+r-1})$ of r successive keystream bits, then he knows that $(K_t, \ldots, K_{t+r-1}) \in X_{z_t^{t+r-1}}$ what can be expressed by $F_{z_t^{t+r-1}}(K_t, \ldots, K_{t+r-1}) = 0$.

Summing up, a general method to find a basis of linearly independent Z-functions for a given (k, ℓ)-combiners is to first generate the sets X_Z and then to compute a basis of annihilators on them with the minimum degree.[2] In the next section, we will first examine the step of computing the sets X_Z and propose several improvements. Afterwards, we will turn our attention to the task of computing annihilators of a given set and discuss several enhancements, including a new quadratic time algorithm that replaces the Gaussian elimination step in previous algorithms and can be implemented easily.

Before we proceed, we want to point out that a similar approach is possible to derive implicit equations for S-boxes used in block ciphers. Such equations were important in the still controversial but nontheless influential attacks on AES presented in [8]. Thus, any improvements in computing annihilators are of independent

[2]Actually, in some cases more direct methods exist to find valid equations (e.g., [1, 7, 9]) but these are not applicable in general and do not guarantee that one has found all equations with the lowest possible degree.

interest in this area and may help to check S-boxes with larger input/output-sizes.

3. Computing the sets X_Z

In this section, we consider the question how to compute the set X_Z for a given (k, ℓ)-combiner and a fixed value $Z \in \mathbb{F}_2^r$. One possibility is to do exhaustive search. That is, compute for all possibilities $(Q, X_1, \ldots, X_r) \in \mathbb{F}_2^{\ell+r\cdot k}$ the value $Z' = f(Q, X_1, \ldots, X_r)$ and keep those values $(X_1, \ldots, X_r)$ with $Z' = Z$. The main effort here is $O(2^{k\cdot r+\ell})$ evaluations of f^r. Of course, if $Z' \neq Z$, it is not necessary to waste this information as it implies that $(X_1, \ldots, X_r) \in X_{Z'}$. Thus, one can compute all 2^r possible sets X_Z at the same time by computing the outputs of $f^r(Q, X_1, \ldots, X_r)$ and sorting $(X_1, \ldots, X_r)$ regarding the output. However, as 2^r different sets X_Z exist, handling these sets simultanously resp. storing them into memory or on hard disk causes additional problems with increasing r. Thus, for large values of r, e.g. $r \geq 10$, one might prefer to generate the sets X_Z independently, meaning that only those values are stored with the correct output, or even to compute these in the same time the annihilators are computed.

In some cases, the effort can be reduced somewhat by exploiting the structure of the (k, ℓ)-combiner. For example, the functions f and Ψ of E_0 depend only on the hamming weight of the inputs X_t but not the concrete values. More precisely, it holds that for all $X_1, \ldots, X_r, Y_1, \ldots, Y_r \in \mathbb{F}_2^4$ with $|X_i| = |Y_i|$, where $|.|$ is the Hamming weight, that

$$f^r(Q, X_1, \ldots, X_r) = f^r(Q, Y_1, \ldots, Y_r) \forall Q \in \mathbb{F}_2^4.$$

Thus, it suffices to consider only the Hamming weights of the inputs, reducing the number of possible inputs from $16 \cdot 16^r$ possible values $(Q, X_1, \ldots, X_r) \in \mathbb{F}_2^4 \times \mathbb{F}_2^{4\cdot r}$ to $16 \cdot 5^r$ possible values $(Q, |X_1|, \ldots, |X_r|) \in \mathbb{F}_2^4 \times \{0, \ldots, 4\}^r$, reducing the number of invocations of f^r accordingly.

Independent of this, one can reduce the number of invocations of f by a kind of divide-and-conquer approach. Herefore, exploit the simple observation that if an input $(X_1, \ldots, X_r) \in \mathbb{F}_2^{k\cdot r}$ cannot lead to a specific out $Z \in \mathbb{F}_2^r$, i.e. $(X_1, \ldots, X_r) \notin X_Z$, then none of the possible extensions $(X_1, \ldots, X_r, X_{r+1}, \ldots, X_s)$ can lead to

any of the outputs $Z||Z' \in \mathbb{F}_2^s$ with $Z' \in \mathbb{F}_2^{s-r}$ and $Z||Z'$ being the concatenation of Z and Z'.

To formalize this, we introduce some more identifiers. The extended update functions $\Psi^r : \mathbb{F}_2^\ell \times \mathbb{F}_2^{r \cdot k} \to \mathbb{F}_2^\ell$ is defined by

$$Q_{r+1} := \Psi^r(Q_1, X_1, \ldots, X_r) \text{ where } Q_{t+1} := \Psi(Q_t, X_t)$$

. That is Ψ^r can be seen as the r-times application of Ψ on the inputs $Q_1, X_1, \ldots, X_r$. For example, for $r = 2$, it holds that

$$\hat{\Psi}(Q_1, X_1, X_2) = \Psi(\Psi(Q_1, X_1), X_2).$$

Definition 3.1. Let a (k, ℓ)-combiner be given and $Z \in \mathbb{F}_2^r$ and $Q \in \mathbb{F}_2^\ell$ be fixed. We introduce the three following sets:

$$\begin{aligned} X_{Q,Z} &:= \{(X_1, \ldots, X_r) \mid f^r(Q, X_1, \ldots, X_r) = Z\} && (3) \\ X_{Z,Q'} &:= \{(X_1, \ldots, X_r) \mid \exists Q : f^r(Q, X_1, \ldots, X_r) = Z \\ &\quad \text{and } \Psi^r(Q, X_1, \ldots, X_r) = Q'\} && (4) \\ X_{Q,Z,Q'} &:= \{(X_1, \ldots, X_r) \mid f(Q, X_1, \ldots, X_r) = Z \\ &\quad \text{and } \Psi^r(Q, X_1, \ldots, X_r) = Q'\} && (5) \end{aligned}$$

These three sets specify again a (sub-)set of all inputs which can lead to the fixed output Z. The difference to X_Z is that additional conditions are imposed. For example, for the elements in $X_{Q,Z}$ it is required that they lead to the output Z if the memory register is set to Q at the beginning of the computations. Similarly, for $X_{Q,Z,Q'}$ it is recommended that the memory state is equal to Q at the beginning and equal to Q' after the computations and that the output is Z.

The sets $X_{Q,Z}$, $X_{Z,Q'}$ and $X_{Q,Z,Q'}$ can be used to compute the sets X_Z iteratively:

Theorem 3.2. *Consider an arbitrary (k, ℓ)-combiner. Then, it holds for all values $Q, Q' \in \mathbb{F}_2^\ell$, $Z \in \mathbb{F}_2^r$ and $Z' \in \mathbb{F}_2^s$:*

$$\begin{aligned} X_Z &= \bigcup_Q X_{Q,Z} = \bigcup_Q X_{Z,Q}, \quad X_{Q,Z||Z'} = \bigcup_{Q'} X_{Q,Z,Q'} \times X_{Q',Z'}, \\ X_{Z||Z'} &= \bigcup_Q X_{Z,Q} \times X_{Q,Z'}, \quad X_{Z||Z',Q'} = \bigcup_Q X_{Z,Q} \times X_{Q,Z',Q'}. \end{aligned}$$

The last equation is important as it implies an iterative way to compute X_Z for $Z \in \mathbb{F}_2^r$ for big values of r. Let for example $Z \in \mathbb{F}_2^r$ and $Z' \in \mathbb{F}_2^{r'}$, then instead of computing $X_{Z||Z'}$ directly with $2^{k\cdot(r+r')}$ invoctations of $f^{r+r'}$, one can compute the sets $X_{Z,Q}$ and $X_{Q,Z'}$ independetly with $2^{k\cdot r}$ executions of f^r and $2^{k\cdot r'}$ executions of $f^{r'}$, respectively. Thus, if $r = r'$, we reduced the effort from $2^{k\cdot 2r}$ invocations of f^{2r} to $2 \cdot 2^{k\cdot r}$ invocations of f^r. Of course, one can reduce the effort further by dividing Z into smaller parts.

4. Computing annihilators - the intersection method

Whereas the computations of the sets X_Z is still doable for average values of r, the calculation of annihilators with a minimum degree is far more time-consuming. This section is on several ways on how to perform this task.

First of all, we need to mention that computing annihilators of degree $\leq d$ for a given set S can be reduced to a problem in linear algebra. Let $S = \{x_1, \ldots, x_s\} \subset \mathbb{F}_2^n$ and $m_1, \ldots, m_{\mu(n,d)}$ be all monomials in n variables of degree $\leq d$. We define a matrix M over $\mathbb{F}_2$ of size $s \times \mu(n,d)$ by setting $M_{i,j} := m_j(x_i)$. Then, it holds that any annihilator $\bigoplus_{j=1}^{\mu(n,d)} c_j \cdot m_j$ of S with a degree $\leq d$ gives a vector $V := (c_1, \ldots, c_{\mu(n,d)})^T$ such that $M \cdot V = \vec{0}$ and vice versa. Thus, a natural way to address this problem is to compute the kernel space of M. This has a time effort in $O(s \cdot \mu(n,d)^2)$. However, in our particular case, it may hold that $s \gg \mu(n,d)$. For example, the sets X_Z for E_0 with $Z \in \mathbb{F}_2^7$ have all a size $\geq 1,200,000$ whereas $\mu(4 \cdot 7, 3) = 3683$. Thus, checking if annihilators of degree ≤ 3 exist over $r = 7$ clocks would need about $1.200,000 \cdot 3683^2 \approx 2^{43.67}$ operations.

Therefore, we propose a different approach. First, $\mu(n,d)$ random points in S are chosen and a matrix M' be calculated, similar to the matrix M described above. More precisely, M' consists of exactly those rows of M which corresponds to the $\mu(n,d)$ chosen points from S. Then, the kernel K' of M' is computed, taking about $\mu(n,d)^3$ operations. In the case of E_0, this value of $\approx 2^{35.54}$. If this kernel consist only the all-zero vector, one knows immediately that no non-trivial annihilators of degree $\leq d$ exist. If the kernel contains other vectors, then each of these vectors defines one function of degree $\leq d$ which is zero on the chosen $\mu(n,d)$

elements. We call them *potential annihilators*. If there are not too many of them, one can check for each potential annihilator if it cancels the remaining $s - \mu(n,d)$ elements in S too. If yes, one has found an annihilator of degree $\leq d$ for the whole set S. If not, one has proven that no such annihilators exist.

Assuming that the number of potential annihilators is too big to be exhaustively checked, then one can pursue as follows. First, one choses another random $\mu(n,d)$ points in S, giving another matrix M'' of size $\mu(n,d) \times \mu(n,d)$. Also here, the kernel K'' is computed, causing an effort of additional $\mu(n,d)^3$ operations. Then, the intersection of both kernels K' and K'' is computed, taking about $\mu(n,d) \cdot (\dim(K') + \dim(K''))^2$ operations which should be significantly less than $\mu(n,d)^3$ in most cases. As the intersection gives now all annihilators for the chosen $2 \cdot \mu(n,d)$ elements, it displays a new set of potential annihilators. From this point, one can proceed as explained above.

Thus, we have seen that the effort is mainly dominated by computing annihilators for sets with $\mu(n,d)$ elements. Using Gaussian elimination here is quite straightforward, but has two disadvantages: First, one has to guess the value of d aforehand, and second, the effort is cubic in $\mu(n,d)$. Thus, one should preferably replace this step by a better method. One candidate is the quadratic time algorithm introduced in [3]. But this algorithm, although very efficient, also needs sophisticated programming methods and careful memory handling.

The intersection method we proposed here permits to avoid its use and permits to take advantage of the fact that the dimension of the monomial basis is far smaller that the dimension of the sets X_Z. In the next Section we will handle the cases of $7, 8$ and 9 clocks with annihilators of degree 3.

5. Experimental results - $7, 8$ and 9 clocks

The case of 6 clocks can be handled easily. Hence in the following we focus on the case of on $7, 8$ and 9 clocks for which we apply the intersection method of the previous section:

• **Research for degree 3 annihilators**

Searching for annihilators of degree 3 we obtain $4 \times r$ variables and a basis of $\mu(4 \times r, d)$ monomials. We construct the 2^r sets X_Z

with size $\approx 2^{r\times 3}$. We do a gaussian elimination in order to find polynomial annihilators of the $\mu(4 \times r, d)$ points, here is 2 cases :

- either the matrix is inversible and there is no annihilator.
- or it isn't and we have a basis of potential annihilators, so we keep this basis in memory and by choosing another set of $\mu(4 \times r, d)$ points we compute a new basis of potential annihilators and we do the intersection method described in Section 4.

clocks	#variables	#monomials	#sets	attempts(average)
7	28	3683	128	2
8	32	5489	256	2
9	36	7807	512	3

clocks	time(1 attempt)	time(total)
7	3 min	12 hours
8	3 min 30	29 hours
9	4 min	102 hours

Applying the method of Section 4 we were able to construct sets of random points which associated matrices had kernels with a null intersection, which proves that no annihilator of degree 3 exist with $7, 8, 9$ clocks. The implementation was made on a PC at 1 Ghz with a program in C.

• **Research for degree** 4 **annihilators**

In that case it is known that there exists at least one annihilator valid for each set. For each set we searched for all possible annihilators. We applied a linear algebra method similar to the previous case but adapted for all points of a set, and we found 4 annihilators for each set we considered: one annihilator common to all the sets and 3 other different annihilators for each set. Our computation was done only for 37 sets but we can conjecture that it is the same for all the sets. Finding 4 annihilators for each set do not permit to reduce the number of bits of streams needed because we can deduce 4 equations of degree 4 from the equation of degree 4 allready known (with 4 consecutive outputs).

clocks	#var.	#mon.	#annihilators	#sets	time(for a set)
7	28	24150	4	128	18 hours

Appendix A. Z-functions and the set X_Z

In this section, we illustrate the concept of Z-functions and the sets X_Z. Herefore, we take a look at the summation generator [22] which is a $(k, \lceil \log_2 k \rceil)$-combiner, based on the integer addition. This means that both the input bits and the memory state are treated as integers and added together. The result forms the output and the next memory state.

More formally, at each clock t there are k input bits $x_{t,1}, \ldots, x_{t,k}$ and the memory state $Q_t \in \mathbb{F}_2^{\lceil \log_2 k \rceil}$. The integer sum of these values is computed, i.e. $S_t := x_{t,1} + \ldots + x_{t,k} + Q_t$ where Q_t is taken as a value in $\{0, \ldots, 2^{\lceil \log_2 k \rceil}\}$. Then, the output and the next memory state Q_{t+1} are computed by

$$z_t := S_t \bmod 2 = (x_{t,1} + \ldots + x_{t,k} + Q_t) \mod 2 \quad (3)$$

$$Q_{t+1} := S_t \operatorname{div} 2 = \left\lfloor \frac{(x_{t,1} + \ldots + x_{t,k} + Q_t)}{2} \right\rfloor \quad (4)$$

For example, in the case $k = 2$ and $\ell = 1$, it holds that

$$z_{t+1} := f(Q_t, x_{t,1}, x_{t,2}) = Q_t \oplus x_{t,1} \oplus x_{t,2} \text{ and} \quad (5)$$

$$Q_{t-1} := \Psi(Q_t, x_{t,1}, x_{t,2}) = Q_t \cdot (x_{t,1} \oplus x_{t,2}) \oplus x_{t,1} \cdot x_{t,2}. \quad (6)$$

We keep now at the example of $k = 2$ and $\ell = 1$. In Table 1, an overview of all possible inputs $(X_1, X_2) \in \mathbb{F}_2^4$ over two clocks and initial memory state $Q \in \mathbb{F}_2$ are given, together with the corresponding outputs $Z \in \mathbb{F}_2^2$. For example, if the initial state of the memory bit is 0 and if the inputs (coming from the LFSRs) are 00 in the first clock and 01 in the second clock, then the output Z of the summation generator is $Z = 01$. Alternatively, one can express this by $f^2(Q, X_1, X_2) = f^2(0, 00, 01) = (0, 1)$.

From Table 1, one can derive directly the four different sets X_Z:[3]

$$\begin{array}{lcl} X_{(0,0)} & = & \{(0000), (0011), (1101), (1110), (0101), (0110), (1001), (1010)\} \\ X_{(0,1)} & = & \{(0001), (0010), (1100), (1111), (0100), (0111), (1000), (1011)\} \\ X_{(1,0)} & = & \{(0100), (0111), (1000), (1011), (0000), (0011), (1101), (1110)\} \\ X_{(1,1)} & = & \{(0101), (0110), (1001), (1010), (0001), (0010), (1100), (1111)\} \end{array} \quad (7)$$

Assume now that we are interested in a Z-function F for $Z = (00)$. That is, F has to be zero on all inputs (X_1, X_2) which can

[3]The commas are removed for the sake of brevity.

Q	0	0	0	0	0	0	0	0
X_1	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
Z	00	01	01	00	10	11	11	10

Q	0	0	0	0	0	0	0	0
X_1	(1,0)	(1,0)	(1,0)	(1,0)	(1,1)	(1,1)	(1,1)	(1,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
Z	10	11	11	10	01	00	00	01

Q	1	1	1	1	1	1	1	1
X_1	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
Z	10	11	11	10	01	00	00	01

Q	1	1	1	1	1	1	1	1
X_1	(1,0)	(1,0)	(1,0)	(1,0)	(1,1)	(1,1)	(1,1)	(1,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
Z	01	00	00	01	11	10	10	11

TABLE 1. All possible input-output combinations over 2 clocks for the summation generator with $k = 2$ inputs and $\ell = 1$ memory bits

lead to the output 00. By definition, this is the set $X_{(0,0)}$. A possible (00)-function therefore is

$$F(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) := x_{1,1} \cdot x_{1,2} \oplus x_{1,1} \oplus x_{1,2} \oplus x_{2,1} \oplus x_{2,2}.$$

Observe that both f and Ψ depend not on the exact value of X but only on its hamming weight. As explained in Section 3, one can exploit this to save memory by only storing only the hamming weights of the elements in X_Z. For example, one can rewrite the sets X_Z in (7) to

$$\begin{array}{rcl} X_{(0,0)} & = & \{[00], [02], [21], [11]\} \\ X_{(0,1)} & = & \{[01], [20], [22], [10], [12]\} \\ X_{(1,0)} & = & \{[10], [12], [00], [02], [21]\} \\ X_{(1,1)} & = & \{[11], [01], [20], [22]\} \end{array} \tag{8}$$

Hereby, $[w_1, w_2]$ denotes the set of all inputs (X_1, X_2) such that the hamming weight of X_1 is equal to w_1 and the same for X_2 and w_2. We denote this kind of set as *blockwise symmetric* as the inputs can be divided into blocks where each block is invariant under permutations.

If one is interested to apply the iterated methods explained at the end of Section 3, one needs to compute the sets $X_{Q,Z}$ and $X_{Z,Q}$. From Table 1, one derive immediately the sets $X_{Q,Z}$. For example, for $Q = 1$ and $Z = 11$, it holds that

$$X_{1,(11)} = \{[01], [20], [22]\}$$

which is naturally a subset of $X_{(11)}$.

References

[1] Frederik Armknecht, Matthias Krause: *Algebraic attacks on Combiners with Memory*, Proceedings of Crypto 2003, LNCS 2729, pp. 162-176, Springer, 2003.

[2] Frederik Armknecht: *Algebraic Attacks and Annihilators*, Proceedings of WEWORC 2005, LNI P-74, pp. 13-21, 2005.

[3] Frederik Armknecht, Claude Carlet, Philippe Gaborit, Simon Knzli, Willi Meier, Olivier Ruatta: *Efficient computation of algebraic immunity for algebraic and fast algebraic attacks*, accepted to Eurocrypt '06.

[4] Bluetooth SIG, *Specification of the Bluetooth system*, Version 1.1, 1 February 22, 2001, available at `http://www.bluetooth.com`

[5] Alex Biryukov, Adi Shamir: *Cryptanalytic Time/Memory/Data tradeoffs for Stream Ciphers*, Proceedings of Asiacrypt 2000, LNCS 1976, pp. 1–13, Springer, 2000.

[6] Vladimor V. Chepyzhov, Ben Smeets: *On A Fast Correlation Attack on Certain Stream Ciphers*, Proceedings of Eurocrypt 1991, LNCS 547 pp. 176–185, Springer, 1991.

[7] Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, Proceedings of Crypto '03, LNCS 2729, pp. 177–194, Springer, 2003.

[8] Courtois, Pieprzyk: *Cryptanalysis of block ciphers with overdefined systems of equations*, Asiacrypt 2002, LNCS 2501, pp. 267–287, Springer, 2002.

[9] Nicolas Courtois, Willi Meier: *Algebraic attacks on Stream Ciphers with Linear Feedback*, Proceedings of Eurocrypt 2003, LNCS 2656, pp. 345–359, Springer, 2003. An extended version is available at `http://www.cryptosystem.net/stream/`

[10] Jean-Charles Faugère: *A new efficient algorithm for computing Gröbner bases (F_4)*, Journal of Pure and Applied Algebra 139, 1-3 (1999), pp. 61–68.

[11] Jean-Charles Faugère, Gwénolé Ars: *An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases*, 2003. Available at `http://www.inria.fr/rrrt/rr-4739.html`

[12] Scott R. Fluhrer, Stefan Lucks: *Analysis of the E_0 Encryption System*, Proceedings of Selected Areas of Cryptography '01, LNCS 2259, pp. 38–48, Springer, 2001.

[13] Jovan Dj. Golic: *Cryptanalysis of Alleged A5 Stream Cipher*, Proceedings of Eurocrypt 1997, LNCS 1233, pp. 239–255, Springer, 1997.

[14] Thomas Johansson, Fredrik Joensson: *Fast Correlation Attacks Based on Turbo Code Techniques*, Proceedings of Crypto 1999, LNCS 1666, pp. 181–197, Springer, 1999.

[15] Thomas Johansson, Fredrik Joensson: *Improved Fast Correlation Attacks on Stream Ciphers via Convolutional Codes*, Proceedings of Eurocrypt 1999, pp. 347–362, Springer, 1999.

[16] Matthias Krause: *BDD-Based Cryptanalysis of Key stream Generators*; Proceedings of Eurocrypt 2002, pp. 222–237, LNCS 2332, Springer, 2002.

[17] Yi Lu, Serge Vaudenay: *Faster Correlation Attack on Bluetooth Keystream Generator E_0*, Proceedings of Crypto 2004, pp. 407–425, LNCS 3152, Springer, 2004.

[18] Yi Lu, Serge Vaudenay: *Cryptanalysis of Bluetooth Keystream Generator Two-Level E_0*, Proceedings of Asiacrypt 2004, pp. 483–499, LNCS 3329, Springer, 2004.

[19] Yi Lu, Willi Meier, Serge Vaudenay: *The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption*, Proceedings of Crypto 2005, pp. 97–117, LNCS 3621, Springer, 2005.

[20] Willi Meier, Enes Pasalic, Claude Carlet: *Algebraic attacks and decomposition of Boolean functions*, Eurocrypt 2004, LNCS 3027, pp. 474–491, Springer, 2004.

[21] Willi Meier, Othmar Staffelbach: *Fast Correlation Attacks on certain Stream Ciphers*, Journal of Cryptology, pp. 159–176, 1989.

[22] Rainer A Rueppel: *Correlation immunity and the summation generator*,Proceedings of Crypto 1985, pp. 260–272, LNCS 218, Springer, 1986.

[23] Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov: *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, Proceedings of Eurocrypt '00, Springer LNCS 1807, pp. 392–407.

[24] Erik Zenner: *On the Efficiency of the Clock Control Guessing Attack*, Proceedings of ICISC 2002, LNCS 2587, Springer, 2002.

[25] Erik Zenner, Matthias Krause, Stefan Lucks: *Improved Cryptanalysis of the Self-Shrinking Generator ACISP 2001*, LNCS 2119, Springer, 2001.

Boolean Functions: Cryptography and Applications BFCA'07
Fonctions Booléennes : Cryptographie & Applications

AN INVERTER ARCHITECTURE FOR ECC-GF(2^m) BASED ON THE STEIN'S ALGORITHM

Maurício Araújo Dias[1] and José Raimundo de Oliveira[2]

Abstract. This paper proposes an architecture which speeds up modular inversion for x coordinate of elliptic curve points. This is a programmable and scalable inverter based on Stein's algorithm. Its configurability is used to make possible optimized circuitry solutions for different elliptic curves, finite fields and algorithms. The inverter can be used with the majority of elliptic curve algorithms over binary finite fields (GF(2^m)). In this work, the m value is 113, 131 or 163. For GF(2^{113}), GF(2^{131}) and GF(2^{163}), the parameters chosen to define the elliptic curves, as well as its points, are defined in the NIST, IEEE P1363, IPSec, WAP, eCheck, ANSI X9.62 and ANSI X9.63 standards. As they represent the smallest standardized finite fields, these m values were chosen in order to avoid the development of excessively large circuits. There is one implementation of the same inverter architecture for each binary finite field chosen for this project. It was developed by using the Altera's Quartus II v5.0 MegaFunctions. The implementations were made in an Altera's EP2S180F1020C4 Stratix II FPGA. By using this inverter for GF(2^{113}), GF(2^{131}) and GF(2^{163}), each modular inversion was computed in, respectively, 324ns, 374ns, and 491ns. To develop this project we considered that high performance has priority over the area occupied by its circuit. Thus, we recommend the use of this circuit in the cases for which no area constraints are imposed but high-performance systems are required.

[1] DCA-FEEC-UNICAMP email: mauricioaraujodias@hotmail.com
[2] DCA-FEEC-UNICAMP email: jro@dca.fee.unicamp.br

J-F. Michon, P. Valarcher, J-B. Yunès (Eds.): BFCA'07

1. Introduction

This work describes the development of an inverter architecture for elliptic curve cryptography over binary finite fields (ECC-GF(2^m)) ([2], [11], [29]). The inverter algorithm is similar to the divider algorithm presented by Wu et al. [16].

We propose a system that works with elliptic curve points, represented by affine coordinates and polynomial basis, implemented by using combinatorial circuits, even though, the current tendency of researches on cryptography implemented in hardware is to propose systems that work with elliptic curve points represented by projective coordinates and normal basis, implemented by using sequential circuits. This tendency aims to get smaller circuits.

Although, our option leads to more complex and larger circuits, the combinatorial implementation increases significantly the performance of the systems compared to sequential implementation used in other systems.

The paper's goal is to introduce a fast combinatorial circuit adequate for commercial FPGAs. It can be used with some ECC-GF(2^m) algorithm, which must also be implemented by hardware.

It must be pointed out that all the algorithms that use elliptic curves over finite fields require the computation of the point $Q = kP$ on E, where k is a large integer and P a point on the elliptic curve E, defined by $y^2 + xy = x^3 + ax^2 + b$ [20]. One method to compute the $Q = kP$ operation is to sweep the binary decomposition of the integer k, doubling on each digit (bit) and adding on digits equal to one ([12], [21]).

The point doublings and the point additions are based on modular additions, multiplications, squaring and divisions [19]. These operations are performed on coordinates of elliptic curve points. To double $P'(P'_X, P'_Y)$ or add the points $P(P_X, P_Y)$ and $P'(P'_X, P'_Y)$ of an elliptic curve, in order to obtain a new point $Q(Q_X, Q_Y)$, it is necessary to compute the following equations [3]:

To double:

$$S = P_X + ((P'_Y)/(P'_X)) \bmod p \quad (1)$$

$$Q_X = (S^2 + S + a) \bmod p \quad (2)$$

$$Q_Y = (S(P_X + Q_X) + P_Y + Q_X) \bmod p \quad (3)$$

To add:

$$S = ((P_Y + P'_Y)/(P_X + P'_X)) \bmod p \quad (4)$$
$$Q_X = (S^2 + S + P_X + P'_X + a) \bmod p \quad (5)$$
$$Q_Y = (S(P_X + Q_X) + P_Y + Q_X) \bmod p \quad (6)$$

In Equations (1) to (6), a is one of the parameters used to define an elliptic curve and p represents the irreducible polynomial.

The Equations (1) to (6) can be optimized in order to achieve a single set of equations. For this purpose, we devise a mix of Equations (1) and (4), (2) and (5), and, finally, (3) and (6), as follows:

To double or add:

$$S = F + ((G + P'_Y)/(H + P'_X)) \bmod p \quad (7)$$
$$Q_X = (S^2 + S + P_X + P'_X + a) \bmod p \quad (8)$$
$$Q_Y = (S(P_X + Q_X) + P_Y + Q_X) \bmod p \quad (9)$$

In Equations (7) to (9), P'_X and P'_Y represent the coordinates of the point that will be doubled or added; P_X and P_Y represent the coordinates of the standard point P, defined in [4]; Q_X and Q_Y represent the coordinates of a new point Q.

Equations (7) to (9) are used to either point doubling or point addition. For point doublings, we have: $F = P_X$, $G = 0$ and $H = 0$, because $P_X = P'_X$ and $P_Y = P'_Y$ for doublings. For point additions, we have: $F = 0$, $G = P_Y$ and $H = P_X$.

Thus, Equations (7) to (9) represent a single set of equations capable to perform either point doubling or point addition using the same logic.

Among all operations that compose Equations (7) to (9), the modular division is the hardest and slowest operation in the finite field arithmetic.

A modular division can be replaced by a modular inversion followed by a multiplication. That is, $((G+P'_Y)/(H+P'_X)) \bmod p$ is equivalent to $((G + P'_Y) \times (H + P'_X) - 1) \bmod p$. It must be pointed out that modular inversion is a particular case of modular division, for which the dividend is a constant value equal to 1.

This way, the algorithm used to perform a modular inversion can be the same algorithm used to compute a modular division with the dividend equal to 1.

Thus, the aforementioned equations are modified as follows:

To double or add:

$$S = F + ((G + P'_Y) \times (H + P'_X) - 1) \bmod p \quad (10)$$

$$Q_X = (S^2 + S + P_X + P'_X + a) \bmod p \quad (11)$$

$$Q_Y = (S(P_X + Q_X) + P_Y + Q_X) \bmod p \quad (12)$$

Despite the challenge to implement a complex and large circuit, the way chosen to make the implementation allows to obtain results that show a high-performance inverter architecture that can be put in a single chip, even for large finite fields.

This observation allows to suppose that architectures working with elliptic curve points represented by projective coordinates and normal basis, when implemented by using combinatorial circuits instead of sequential circuits, allow to achieve a better balance between performance and area results.

The remainder of the paper is organized as follows. Section 2 summarizes related work. Section 3 briefly explains the use of Stein's Algorithm to perform a modular inversion. The use of Altera Mega-Functions to implement an optimization of the Stein's algorithm is described in Section 4. Section 5 provides a brief description of the inverter implementations in a FPGA. The results achieved by these implementations are presented in Section 6. Finally, Section 7 contains the conclusions.

2. Related Work

Researchers have been working in the development of new algorithms, in order to achieve less complex and faster modular divisions and inversions. They have based their works on three different methods to perform modular divisions and inversions over GF(2^m): Little Fermat's Theorem, Gaussian Elimination and Greatest Common Divisor (GCD).

In late 80's, Itoh and Tsujii [10] proposed an algorithm based on Little Fermat's Theorem to compute the modular inversion. Their algorithm used normal basis instead of polynomial basis.

In 1992, Hasan and Bhargava [9] presented a bit-serial divider, based on Gaussian Elimination applied to systems of linear equations.

Albeit the improvement achieved by these two works, our implementations of these algorithms generate excessively large and

slow combinatorial circuits; thus, the two methods do not meet the features expected to our work.

A third method yields better results. The works that use this method are commented next.

From 2001 to 2003, Eberle, Gura, Shantz, Gupta, et al. describe implementations of iterative transformations of the GCD based on Euclid's algorithm ([6], [7], [13]).

An implementation of our inverter using the Euclid's algorithm achieves high performance results but the area of the circuit is still larger than that of available commercial FPGAs.

Alternatively, it is possible to work with the GCD method based on the Stein's algorithm. The first description of this algorithm was in 1967 by Stein [14]. In 2004, Wu et al. [16] published an important optimization of the Stein's algorithm, designed to avoid its complex degree comparisons of polynomials. Recently, Dormale and Quisquater [5] presented optimizations for this algorithm using bit-serial implementations. These three papers are the bases of this work.

3. The Stein's Algorithm

Our implementations of all aforementioned algorithms allow us to conclude that the Stein's method for finding the Greatest Common Divisor of two integers is the best choice for calculating the modular inversion for this work. It is important to define which implementation of the Stein's method better satisfies the speed and area requirements for this project. A variety of optimizations have been proposed ([5], [8], [15], [16], [17], [18]) based on Stein's algorithm [14]. The Algorithms 3.1 and 3.2 describe two of these optimizations.

In Algorithm 3.1, P'_X and p represent, respectively, the value that will be inverted and the irreducible polynomial. The `deg()` function returns the polynomial degree. The + and /2 operations can be understood, respectively, as `xor` and one bit right shift.

It must be pointed out that, albeit the optimizations, Algorithm 3.1 still presents the operation if $\texttt{deg}(A) \geq \texttt{deg}(B)$, which is a degree comparison of polynomials. This operation's complexity decreases the algorithm performance.

Algorithm 3.2 is similar to the divider algorithm designed by Wu et al. [16] to avoid the complex degree comparisons of polynomials.

Algorithm 3.1.

$(A,\ B) \leftarrow (P'_X,\ p)$
$(U,\ V) \leftarrow (1,\ 0)$

while $A \neq 0$ and $B \neq 1$
  if $A_0 = 1$
    if $\deg(A) \geq \deg(B)$
      $(A,\ B) \leftarrow (A + B,\ U + V)$
    else
      $(A,\ B) \leftarrow (A + B,\ A)$
      $(U,\ V) \leftarrow (U + V,\ U)$
    endif
  endif
  $A \leftarrow A/2$
  $U \leftarrow (U/2) \bmod p$
endwhile

Algorithm 3.2.

$(A,\ B,\ U,\ V) \leftarrow (P'_X,\ p,\ 1,\ 0)$
$(DCC,\ FLAG,\ slice) \leftarrow (2,\ 1,\ 2m - 1)$

while $slice > 0$
  if $A_0 = 1$
    if $FLAG = 1$ and $DCC_0 = 0$
      $(A,\ B) \leftarrow (A + B,\ A)$
      $(U,\ V,\ FLAG) \leftarrow (U + V,\ U,\ 0)$
    else
      $(A,\ B) \leftarrow (A + B,\ U + V)$
    endif
  endif
  $(A,\ U) \leftarrow (A/2,\ (U/2) \bmod p)$
  if $FLAG = 0$ and $DCC_0 = 0$
    $DCC \leftarrow DCC/2$
  else
    $(DCC,\ FLAG) \leftarrow ((DCC \times 2),\ 1)$
  endif
  $slice \leftarrow slice - 1$
endwhile

In Algorithm 3.2, the degree comparison of polynomials is replaced by FLAG and DCC variables. It allows Algorithm 3.2 to achieve the same results faster than Algorithm 3.1. Algorithm 3.2 needs at most $2m - 1$ iterations to perform a modular inversion.

4. The Combinatorial Circuit

Algorithm 3.2 represents the bases of the combinatorial circuit presented in this paper. When its code is represented using a schematic, it constitutes a slice, which is exactly equal to a single iteration of the Algorithm 3.2. Whereas Algorithm 3.2 requires at most $2m-1$ iterations to perform a modular inversion, we need to link serially $2m-1$ slices in order to create a combinatorial circuit of modular inversion.

All slices are identical. Each slice has six inputs and six outputs: A_{in}, B_{in}, U_{in}, V_{in}, DCC_{in}, A_{out}, B_{out}, U_{out}, V_{out} and DCC_{out} (all of them are $m + 1$ bits wide); $FLAG_{in}$ and $FLAG_{out}$ (both are 1 bit wide). The outputs of the first slice are connected with the inputs of the second slice; the outputs of the second slice are connected with the inputs of the third slice and so on.

The inputs of the first slice are started as follows: $(A_{in},\ B_{in},\ U_{in},\ V_{in},\ DCC_{in},\ FLAG_{in}) \leftarrow (P'_X,\ p,\ 1,\ 0,\ 2,\ 1)$. A few hundreds

of nanoseconds later, the output V_{out} presents the modular inverse of P'_X in the last slice.

The schematic of a slice is shown by figures and briefly described in the remainder of this section. Each gate represents graphically a Quartus II Mega-Function.

Figure 1 shows the so called "control unit" of the slice. This "control unit" has logical gates controlling the AUX and $FLAG_{out}$ signals, which are responsible for determining the dataflow among the slice components.

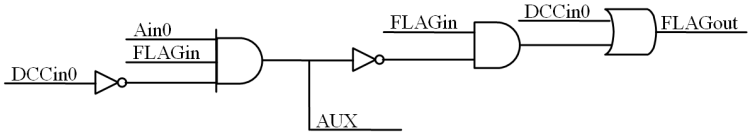


FIGURE 1. $FLAG$ and AUX handling.

The multiplexer in Figure 2 represents the DCC counter in the slice. When $FLAG_{out}$ is 1, the DCC_{in} value is left shifted one bit; when $FLAG_{out}$ is 0, the DCC_{in} value is right shifted one bit (as a Johnson counter).

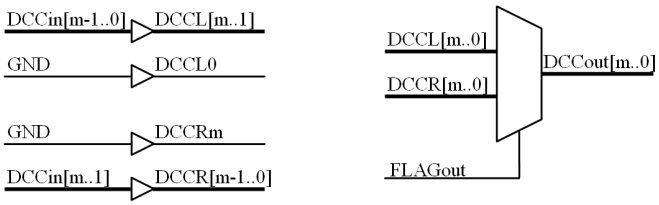


FIGURE 2. DCC handling.

Figure 3 shows that A_{in} `xor` B_{in} operation occurs according to A_{in0}, in order to generate A_{out}. Before the `xor` operation, A_{in} and B_{in} are both right shifted one bit.

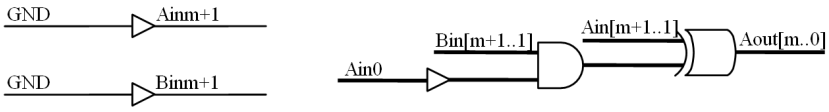


FIGURE 3. A handling.

Figure 4 shows that U_{in} `xor` V_{in} operation occurs according to A_{in0}, in order to generate the intermediate value U. The U and p values are both right shifted one bit. Once shifted, they are both submitted to an `xor` operation in order to generate U_{out}.

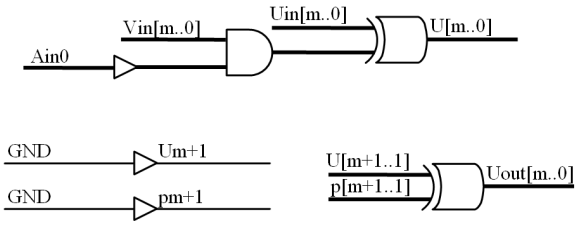


FIGURE 4. U handling.

When AUX is 0, the B_{in} value is placed in the B_{out} output, without any change. Otherwise, the B_{out} output presents the A_{in} value, as shown in Figure 5.

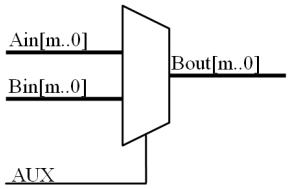


FIGURE 5. B handling.

As presented in Figure 6, the V_{in} input is placed in the V_{out} output when AUX is 0. Otherwise, the U_{in} goes to the V_{out} output.

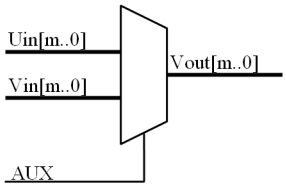


FIGURE 6. V handling.

Figures 1 to 6 compose the schematic of any of the $2m-1$ slices that constitute the combinatorial circuit proposed in this work.

5. Implementations

The schematic of our circuit was developed, compiled and simulated using the Quartus II software v5.0 for Altera's FPGA [1]. This software was executed on a PC Pentium 4, running at 3.2GHz, with 1GB of RAM and 30GB of HD.

There is one implementation for each finite field (GF(2^m)) chosen for this work. In this project, the m value is 113, 131 or 163.

These implementations allow to evaluate how much time the circuit needs to calculate a modular inversion for each of the three different inverter implementations.

For GF(2^{113}), GF(2^{131}) and GF(2^{163}), the parameters chosen to define the elliptic curves, as well as its points, are defined in the NIST, IEEE P1363, IPSec, WAP, eCheck, ANSI X9.62 and ANSI X9.63 standards [4].

In order to insert this combinatorial circuit in a more practical context, we have developed a cryptosystem that can be implemented, for example, in the form of a PC-board adapter.

Figure 7 shows how our combinatorial circuit for modular inversion can be implemented in the PC-board adapter in order to compose this cryptosystem.

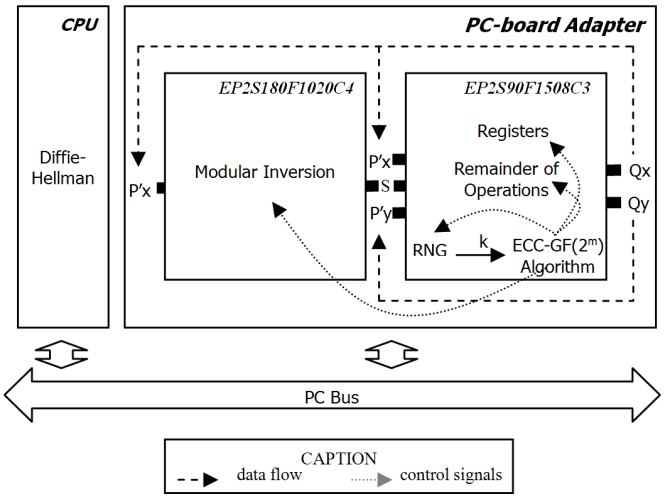


FIGURE 7. Basic diagram of the PC-board adapter.

The same Figure 7 presents all the basic diagram of the PC-board adapter, used to exemplify a possible implementation of

the cryptosystem. It shows the on-board elements and the PC's components that communicate with the adapter.

The cryptosystem was implemented in the EP2S180F1020C4 and EP2S90F1508C3 FPGAs, due to high performance and density requirements. The first implements only the modular inversion; the latter implements the remainder of operations presented in Equations (10) to (12).

Moreover, the second FPGA also implements a cryptographic algorithm, a random number generator (RNG), some general purpose registers and the logic of the bus interface. All these elements are commented next.

This cryptosystem does not work with a specific algorithm. Any ECC-GF(2^m) algorithm can be described in VHDL and implemented in the FPGA. The algorithm must generate some signals in order to control the point doublings and the point additions operations. It must also work with elliptic curve points represented by affine coordinates and polynomial basis.

The adapter does not work with an exclusive random number generator. On the contrary, it allows the implementation of different types of random number generators.

The implementation of the general purpose registers must also be flexible. It allows to define the width of the registers according to the chosen finite field.

Finally, the bus interface must be described and implemented according to the type of bus used in the PC (PCI, AGP or other).

The cryptosystem uses our inverter architecture and these other elements present in the adapter in order to perform the $Q = kP$ operation.

In a practical example, a program that implements the Diffie-Hellman key-exchange model is executed by the CPU of a PC. The model needs a point Q, calculated by using the $Q = kP$ equation. In order to achieve a better performance, the program calls our cryptosystem to perform the $Q = kP$ operation. When the adapter starts working, the program sends the point P' to the adapter through the PC bus. In the adapter, the private key k is generated by the random number generator. The cryptosystem uses the private key k and the point P' in order to calculate $Q = kP'$, according to the cryptographic algorithm logic, that will use our combinatorial circuit. At the end of the process, the adapter sends the Q point back to the program through the PC bus. With the help of our cryptosystem, this program can obtain

the point $Q = kP$ in a few hundreds of microseconds, as it will be presented in next section.

As a general rule, the implementations and tests done aim to obtain results to show how our combinatorial circuit can improve the ECC.

6. Results

The results achieved with the aforementioned implementations are shown in Table 1, 2, 3 and 4.

Table 1 presents the results of the simulation of the inverter architecture for GF(2^{113}), GF(2^{131}) and GF(2^{163}) implementations. Each line in the first column identifies a binary finite field used in that implementation. In the remainder of the columns, the rows refer to each of these finite fields. The second column shows the number of pins used in the FPGA. The third column presents the number of slices serially linked in the circuit. The area of the circuit is represented in number of LUTs (FPGA's look-up tables) in Column 4. The circuit delay, presented in Column 5, informs how many nanoseconds the circuit needs to compute a modular inversion. Finally, Column 6 shows how many days, hours, minutes and seconds are necessary to compile the inverter architecture.

Finite Field	Number of Pins	Number of Slices	Number of LUTs	Circuit Delay (ns)	Compilation Time (dd:hh:mm:ss)
113	229	225	60,361	324	01:18:41:19
131	265	261	82,082	374	03:04:51:36
163	329	325	128,265	491	10:03:31:41

TABLE 1. Results of three implementations of the inverter architecture.

Table 2 shows the performance of different implementations for modular inversion. The first column identifies three different implementations with their respective references. Column 2 shows the time (in microseconds) necessary to calculate a modular inversion for each implementation.

Based on the results shown in Column 2, it is possible to conclude that our combinatorial circuit for modular inversion presents high performance.

Implementations	Modular Inversion Time (μs)
Ext. Euclides [28]	2.509
Itho-Tsujii [28]	0.760
Our Inverter	0.490

TABLE 2. Results of three implementations for modular inversion.

Table 3 shows the performance of different software or hardware (S/H) implementations. There are implementations working with elliptic curve points represented by either affine or projective coordinates and either polynomial or normal bases.

Implementations	S/H	Finite Fields	Plataforms	$Q = kP$ (ms)
Montgomery [22]	S	GF(2^{163})	UltraSparc 64-bit	13.50
Almost Inv. [23]	S	GF(2^{155})	DEC Alpha 64-bit	7.80
Coprocessor [24]	H	GF(2^{155})	VLSI (ASIC)	3.90
Coprocessor [25]	H	GF(2^{155})	Xilinx XC4020XL	18.40
ECP [26]	H	GF(2^{167})	Xilinx XCV400E	0.21
Montgomery [27]	S	GF(2^{163})	Sun Fire 280R	3.11
Cryptographic Processor [27]	H	GF(2^{163})	Xilinx Virtex-II XCV2000E-7	0.14
Our Cryptosystem	H	GF(2^{163})	Altera Stratix II EP2S180F1020C4 EP2S90F1508C3	0.10

TABLE 3. Performance of hardware/software implementations.

The first column identifies eight different implementations listed on the table with their respective references. In the remainder of the columns, the rows refer to each of these implementations. Column 2 classifies each row as a software (S) or hardware (H) implementation. The finite fields are presented in the third column. Column 4 presents the platforms used to develop each of the implementations. Finally, Column 5 shows the time (in milliseconds) necessary to calculate $Q = kP$ for each implementation.

Based on the results presented in Column 5, it is possible to conclude that our cryptosystem, using our combinatorial circuit

for modular inversion, can perform the $Q = kP$ operation significantly fast.

Table 4 presents results achieved by executing a program, which implements the Diffie-Hellman key-exchange model, with or without the help of our cryptosystem. The program was executed on a PC Pentium 4, running at 3.2GHz, with 1GB of RAM and 30GB of HD.

	Diffie-Hellman	
Finite Field	Without our cryptosystem (s)	With our cryptosystem (μs)
113	1	125
131	2	161
163	5	244

TABLE 4. Time necessary to execute a program with or without our cryptosystem.

Each row in Table 4 refers to a binary finite field presented in the first column. Column 2 shows the time, in average, necessary to execute the program without our cryptosystem. Column 3 presents the time, in average, necessary to execute the same program with the help of our cryptosystem.

Based on the results presented in Table 4, it is possible to conclude that the program execution with the help of our cryptosystem is significantly faster than the program execution without the help of our cryptosystem.

7. Conclusions

This paper has introduced an ECC-GF(2^m) inverter architecture, based on Stein's algorithm. It is recommended for high speed but with no area constraints cryptographic systems, which work with affine coordinates and polynomial basis.

The performance of the inverter was demonstrated by using three different implementations done in Altera's EP2S180F1020C4 Stratix II, each of which for GF(2^{113}), GF(2^{131}) and GF(2^{163}).

By using this inverter architecture for GF(2^{113}), GF(2^{131}) and GF(2^{163}), the results achieved for each modular inversion are, respectively, 324ns, 374ns and 491ns.

The aforementioned implementations meet two key features: programmability and configurability. The first feature allows implementations to be programmed to work with more efficient ECC-GF(2^m) algorithms in the future. The second feature allows to update the inverter performance and size to meet future requirements, by reconfiguration.

References

[1] Altera Corporation. *Quartus II, Programmable Logic Development System & Software*. Data Sheet, ver. 1.01, 1999.

[2] I. Blake, G. Seroussi and S. Nigel. *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.

[3] Certicom Research. *SEC 1: Elliptic Curve Cryptography*, Standards for Efficient Cryptography, Version 1.0, `www.certicom.com`, 2000.

[4] Certicom Research. *SEC 2: Recommended Elliptic Curve Domain Parameters*, Standards for Efficient Cryptography, Version 1.0, `www.certicom.com`, 2000.

[5] G. M. Dormale, J-J. Quisquater. *Novel Iterative Digit-Serial Modular Division over GF(2^m)*. `http://www.dice.ucl.ac.be/crypto/files/publications/pdf254.pdf`, 2005.

[6] H. Eberle, N. Gura, S. C. Shantz and V. Gupta. *A Cryptographic Processor for Arbitrary Elliptic Curves over GF(2^m)*. Sun Microsystems Laboratories, SMLI TR-2003-123, May 2003.

[7] N. Gura, S. C. Shantz, H. Eberle, S. Gupta, V. Gupta, D. Finchelstein, E. Goupy, D. Stebila. *An End-to-End Systems Approach to Elliptic Curve Cryptography*. CHES 2002, LNCS 2523, pp. 349-365, 2002.

[8] J. Goodman and A. P. Chandrakasan. *An Energy-Efficient Reconfigurable Public-Key Cryptography Processor*. IEEE J. Solid-State Circuits, vol. 36, pp. 1808-1820, Nov. 2001.

[9] M. A. Hasan and V. K. Bhargava. *Bit-Serial Systolic Divider and Multiplier for Finite Fields GF(2^m)*. IEEE Transaction on Computers, vol. 41, no. 8, pp. 972-980, 1992.

[10] T. Itoh and S. Tsujii. *A Fast Algorithm for Computing Multiplicative Inverses in GF(2^m) Using Normal Bases*. Information and Computation, vol. 78, pp. 171-177, 1988.

[11] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.

[12] IEEE P1363/D13 (Draft Version 13). *Standard Specification for Public Key Cryptography*. IEEE Inc., 1999.

[13] S. C. Shantz. *From Euclid's GCD to Montgomery Multiplication to the Great Divider*. Sun Microsystems Laboratories, SMLI TR-2001-95.

[14] J. Stein. *Computational problems associated with Racah algebra*. J. Computational Physics, vol. 1, pp. 397-405, 1967.

[15] N. Takagi. *A VLSI Algorithm for Modular Division Based on the Binary GCD Algorithm*. IEICE Trans. Fundamentals, vol. E81-A, pp. 724-728, May 1998.

[16] C. H. Wu, C. M. Wu, M. D. Shieh and Y. T. Hwang. *High-Speed, Low-Complexity Systolic Designs of Novel Iterative Division Algorithms in GF(2^m)*. IEEE Trans. on Computers, vol. 53, no. 3, pp. 375-380, 2004.

[17] Y. Watanabe, N. Takagi, and K. Takagi. *A VLSI Algorithm for Division in GF(2^m) Based on Extended Binary GCD Algorithm*. IEICE Trans. Fundamentals, vol. E85-A, pp. 994-999, May 2002.

[18] Y. Watanabe and N. Takagi. *A VLSI Algorithm for Division on GF(2^m) Based on Binary Method*. Proc. 2000 Eng. Sciences Soc. Conf. IEICE, A-3-15, p. 82, Sept. 2000

[19] E. D. Win, A. Bosselaers, S. Vandenberghe, P. D. Gersem and J. Vandewalle. *A Fast Software Implementation for Arithmetic Operations in GF(2^n)*. Advances in Cryptology - ASIACRYPT'96, Lecture Notes in Computer Science 1163, pp.65-76, Springer-Verlag, 1996.

[20] F. Morain and J. Olivos. *Speeding up the Computations on an Elliptic Curve Using Addition-Subtraction Chains*. Rapport INRIA, issue 983, March 1989.

[21] P. Guillot and O. Orcière. *Speeding up Elliptic Curve Computations Using Addition-Substraction Chains and Horner Rule*. Thomson-CSF Communications, March 4, 1998.

[22] J. López and R. Dahab. *Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation*. Workshop on Cryptographic Hardware and Embedded Systems (CHES) 1999, LNCS 1717, pp. 316-327g, 1999.

[23] R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck. *Fast Key Exchange with Elliptic Curve Systems*, In D. Coppersmith, editor, Advances in Cryptography, Crypto 95, volume LNCS 963, Springer-Verlag, 1995.

[24] G. B. Agnew, R. C. Mullin and S. A. Vanstone. *An Implementation of Elliptic Curve Cryptosystems over F2155*, IEEE Journal on Selected Areas in Communications, 11(5):804-813, 1993.

[25] S. Sutikno, R. Effendi and A. Surya. *Design and Implementation of Arithmetic Processor F2155 for Elliptic Curve Cryptosystems*, The 1998 IEEE Asia-Pacific Conference on Circuits and Systems, pages 647-650.

[26] G. Orlando and C. Paar. *A High-Performance Reconfigurable Elliptic Curve Processor for GF(2^m)*. Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2000, LNCS 1965, pp. 41-56, 2000.

[27] H. Eberle, N. Gura, S. C. Shantz and V. Gupta. *A Cryptographic Processor for Arbitrary Elliptic Curves over GF(2^m)*. Sun Microsystems Laboratories, SMLI TR-2003-123, May 2003.

[28] M. A. N. Cervantes, K. P. G. Avila and F. R. Henriquéz. *Inversion Modular en Campos Finitos Binarios*. Centro de Investigación y Estudios Avanzados del IPN, DIE-SC, 2006.

[29] D. Hankerson, A. Menezes and S. Vanstone. *Guide to Elliptic Curve Cryptography*, Springer Professional Computing, 2004.

Boolean Functions: Cryptography and Applications BFCA'07
Fonctions Booléennes : Cryptographie & Applications

COMPUTING MÖBIUS TRANSFORMS OF BOOLEAN FUNCTIONS AND CHARACTERISING COINCIDENT BOOLEAN FUNCTIONS

Josef Pieprzyk[1] and Xian-Mo Zhang[1]

Abstract. The Möbius transform of Boolean functions is often involved in cryptographic design and analysis. This work is composed of two parts. In the first part we compute Möbius transform by different methods and study its cryptographic properties. In the second part we focus on the special case when a Boolean function is identical with its Möbius transform. We call such functions coincident. We further characterise coincident functions and study their cryptographic properties.

Key Words: Boolean Functions, Möbius transform, Coincident Functions

1. Introduction to This Work

Recent developments in algebraic analysis of ciphers put an emphasis on methods and techniques that treat a cryptographic system as a collection of Boolean functions, describe them by their algebraic normal forms (ANFs), and then examine their algebraic properties such as sparseness, algebraic degree, number of overdefined relations, number of monomials, etc. A prerequisite for an efficient algebraic analysis is the ability to represent Boolean functions and their relations by their short algebraic forms. Moreover, most of time the designers of Boolean functions are working with their truth tables and the translation from a truth table to its

[1] Centre for Advanced Computing - Algorithms and Cryptography, Department of Computing, Macquarie University, Sydney , NSW 2109, Australia, email: josef,xianmo@ics.mq.edu.au

unique algebraic normal form (ANF) is not immediate. The aim of this work is to investigate the Möbius transform and its significance in Cryptography. What is interesting about the Möbius transform is that it allows to define a class of Boolean functions whose ANFs can be written easily from their truth tables (and vice versa). This nice property can be useful for analysis and design of small cryptographic S-boxes. More importantly, it could be used to justify security level of a cryptographic scheme if its truth table is big enough so it is impossible to construct its truth table and as the results, it is impossible to determine its ANF.

It is a well-known fact that a Boolean function f of n variables $(x_1, \ldots, x_n)$ can be uniquely represented a polynomial in Formula (1) where g is also a function of n variables that characterises f. We call g the Möbius transform of f. In this work we denote this relation by $g = \mu(f)$. We present methods to compute $\mu(f)$ and study cryptographic properties of $\mu(f)$. We further propose the concept of coincident functions. A Boolean function f is called coincident if f is identical with $\mu(f)$. We consider an example, $f(x_1, x_2, x_3) = x_3 \oplus x_2 \oplus x_1 \oplus x_1x_3 \oplus x_1x_2x_3$. From the ANF of f, we know the truth table of $\mu(f)$: (01101101). On the other hand, by computing, we know the truth table of f: (01101101). Then f is a coincident function on $(GF(2))^3$. In general we can obtain the ANF/truth table of a coincident function from its truth table/ANF without computing. We characterise the coincident functions and examine their cryptographic properties.

2. Introduction to Boolean Functions

Throughout the paper we use the following notations. The vector space of n-tuples of elements from $GF(2)$ is denoted by $(GF(2))^n$. We write all vectors in $(GF(2))^n$ as $(0, \ldots, 0, 0) = \alpha_0$, $(0, \ldots, 0, 1) = \alpha_1$, ..., $(1, \ldots, 1, 1) = \alpha_{2^n-1}$, and call α_i the *binary representation* of integer i, $i = 0, 1, \ldots, 2^n - 1$. A Boolean function f is a mapping from $(GF(2))^n$ to $GF(2)$ or simply, a function f on $(GF(2))^n$. We write f more precisely as $f(x)$ or $f(x_1, \ldots, x_n)$ where $x = (x_1, \ldots, x_n)$. The *truth table* of a function f on $(GF(2))^n$ is a $(0, 1)$-sequence defined by $(f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{2^n-1}))$, The *Hamming weight* of a (0, 1)-sequence ξ, denoted by $HW(\xi)$, is defined as the number of nonzero coordinates of ξ. In particular, if ξ is the truth table of a function f, then $HW(\xi)$

is called the *Hamming weight* of f, denoted by $HW(f)$. f is said to be *balanced* if $HW(f) = 2^{n-1}$. The *Hamming distance* between functions f and g on $(GF(2))^n$, denoted by $d(f,g)$ is defined as $d(f,g) = HW(f \oplus g)$. The function f can be uniquely represented by a polynomial

$$f(x_1,\ldots,x_n) = \bigoplus_{\alpha\in(GF(2))^n} g(a_1,\ldots,a_n)x_1^{a_1}\cdots x_n^{a_n} \qquad (1)$$

where $\alpha = (a_1,\ldots,a_n)$, and g is also a function on $(GF(2))^n$, called the ***Möbius transform*** of f. The polynomial representation of f is called the *algebraic normal form* (ANF) of the function f and each $x_1^{a_1}\cdots x_n^{a_n}$ is called a *monomial (term)* in the ANF of f. The *algebraic degree*, or *degree*, of f, denoted by $deg(f)$, is defined as $deg(f) = \max_{(a_1,\ldots,a_n)} \{HW(a_1,\ldots,a_n) \mid g(a_1,\ldots,a_n) = 1\}$. f is called *affine* if its ANF has the following form: $f(x) = a_1x_1 \oplus \cdots \oplus a_nx_n \oplus c$ where $x = (x_1,\ldots,x_n)$, $a_1,\ldots,a_n,c \in GF(2)$ are constant. In particular f is called *linear* if $c = 0$.

3. Computing $\mu(f)$ by Matrix

Notation 1. *Let $\mathcal{R}_n$ denote the set of all functions on $(GF(2))^n$. In this work we write $\mu(f) = g$ where g is the* Möbius transform *of f, defined in Formula (1).*

By definition, it is easy to verify that the Möbius Transform μ is a one-to-one linear mapping from $\mathcal{R}_n$ to $\mathcal{R}_n$.

Notation 2. *We define $2^n \times 2^n$ (0, 1)-matrix, denoted by T_n: the ith row of T_n ($n \geq 1$) is the truth table of $x_1^{a_1}\cdots x_n^{a_n}$ where $(a_1,\ldots,a_n)$ is the binary representation of the integer i. In addition, we define $T_0 = 1$.*

Theorem 3.1. *T_n, defined in Notation 2, satisfies $T_s = \begin{bmatrix} T_{s-1} & T_{s-1} \\ O_{2^{s-1}} & T_{s-1} \end{bmatrix}$, where $O_{2^{s-1}}$ denotes the $2^{s-1} \times 2^{s-1}$ zero matrix, $s = 1, 2, \ldots$.*

Proof. We prove the theorem by induction on n. By definition, the 0th row of T_1 is the truth table of the constant function $f(x_1) = x_1^0 = 1$ and the 1st row of T_1 is the truth table of the function $f(x_1) = x_1$. Then $T_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Recall Notation 2 where $T_0 = 1$.

Then the theorem holds for $n = 1$. Assume that the lemma is true when $1 \leq n \leq s-1$. Let $n = s$. Consider the monomial $x_1^{a_1} \cdots x_s^{a_s}$. There exist two cases to be considered: $a_1 = 0$ (Case 1) and $a_1 = 1$ (Case 2). In Case 1 $x_1^{a_1} \cdots x_s^{a_s} = x_2^{a_2} \cdots x_s^{a_s}$. By the induction assumption, the ith row of T_{s-1} is the truth table of $x_2^{a_2} \cdots x_s^{a_s}$. Due to the relation between T_s and T_{s-1}, it is easy to verify that the ith row of T_s is the truth table of $x_1^{a_1} x_2^{a_2} \cdots x_s^{a_s}$ with $a_1 = 0$. In Case 2 $x_1^{a_1} \cdots x_s^{a_s} = x_1 x_2^{a_2} \cdots x_s^{a_s}$. Due to the relation between T_s and T_{s-1}, it is easy to verify that the ith row of T_s is the truth table of $x_1^{a_1} x_2^{a_2} \cdots x_s^{a_s}$ with $a_1 = 1$. □

Example 3.2. By using Theorem 3.1, we can construct T_1, T_2, T_3, ... T_1 has two rows (1 1) and (0 1). T_2 has four rows (1 1 1 1), (0 1 0 1), (0 0 1 1) and (0 0 0 1). T_3 has eight rows: (1 1 1 1 1 1 1 1), (0 1 0 1 0 1 0 1), (0 0 1 1 0 0 1 1), (0 0 0 1 0 0 0 1), (0 0 0 0 1 1 1 1), (0 0 0 0 0 1 0 1), (0 0 0 0 0 0 1 1) and (0 0 0 0 0 0 0 1). It is noted that (1, 0, 1) is the binary representation of integer 5. By the definition of T_n, the 5th row of T_3, (0 0 0 0 0 1 0 1), is the truth table of $x_1^1 x_2^0 x_3^1 = x_1 x_3$.

Lemma 3.3. *(i) $T_s^2 = I_{2^s}$ where I_{2^s} is the $2^s \times 2^s$ identity matrix, (ii) $(T_s \oplus I_{2^s})^2 = 0_{2^s}$, (iii) $T_s(T_s \oplus I_{2^s}) = (T_s \oplus I_{2^s})T_s = T_s \oplus I_{2^s}$, where $s = 1, 2, \ldots$.*

Proof. (i) can be proved by induction. (ii) and (iii) are immediate consequences of (i). □

Theorem 3.4. *Let f and g be functions on $(GF(2))^n$. Denote the truth tables of f and g by ξ and η respectively. Then the following statements are equivalent: (i) $g = \mu(f)$, (ii) $f = \mu(g)$, (iii) $\eta T_n = \xi$, (iv) $\xi T_n = \eta$.*

Proof. Assume that (i) holds. We now prove (iii). We recall that $\eta = (g(\alpha_0), g(\alpha_1), \ldots, g(\alpha_{2^n-1}))$. By the definition of T_n, ηT_n is the truth table of f. Then $\eta T_n = \xi$. This proves (iii). Assume that (iii) holds. Let $g' = \mu(f)$ and η' be the truth table of g'. Since we have proved (i) $\Longrightarrow$ (iii), we know that $\eta' T_n = \xi$. Comparing $\eta' T_n = \xi$ with $\eta T_n = \xi$, since T_n is invertible, we know $\eta' = \eta$ and then $g' = g$. We then have proved (i). Therefore (i) $\Longleftrightarrow$ (iii). Symmetrically, (ii) $\Longleftrightarrow$ (iv). Due to (i) of Lemma 3.3, (iii) $\Longleftrightarrow$ (iv). The proof is completed. □

It is noted that the equivalence between (i) and (ii) of Theorem 3.4 was previously proved in [3]. However we regain it here by

using a different concept. Theorem 3.4 enables us to compute the truth table/ANF from the ANF/truth table of a function by using the matrix T_n.

Example 3.5. Assume that we know the ANF of f on $(GF(2))^3$: $f(x_1, x_2, x_3) = 1 \oplus x_2 \oplus x_1 \oplus x_2x_3 \oplus x_1x_2x_3$. Set $g = \mu(f)$. From the ANF of f, we know that g has the truth table (10111001). By using Theorem 3.4, $(10111001)T_3 = (11010011)$ is the truth table of f. Conversely, assume that we know the truth table of function f on $(GF(2))^3$: (11010011). By using Theorem 3.4, $(11010011)T_3 = (10111001)$ is the truth table of the $\mu(f)$. Therefore we obtain the ANF of f: $f(x_1, x_2, x_3) = 1 \oplus x_2 \oplus x_1 \oplus x_2x_3 \oplus x_1x_2x_3$.

Theorem 3.6. *μ^2 is the identity transformation, or, $\mu^{-1} = \mu$.*

Proof. The theorem is true due to (i) of Lemma 3.3. □

4. Computing $\mu(f)$ by Polynomials

Notation 3. *For any $\alpha \in (GF(2))^n$, we define a function D_α on $(GF(2))^n$ as follows: $D_\alpha(x) = (1 \oplus a_1 \oplus x_1) \cdots (1 \oplus a_n \oplus x_n)$ where $x = (x_1, \ldots, x_n)$, $\alpha = (a_1, \ldots, a_n)$.*

Furthermore, it is known that for any function f on $(GF(2))^n$, we have

$$f(x) = \bigoplus_{\alpha \in (GF(2))^n} f(\alpha) D_\alpha(x) \tag{2}$$

For any two functions f and f' on $(GF(2))^n$,
$f(x) \oplus f'(x) = \bigoplus_{\alpha \in (GF(2))^n} (f(\alpha) \oplus f'(\alpha)) D_\alpha(x)$ and
$f(x) \cdot f'(x) = \bigoplus_{\alpha \in (GF(2))^n} (f(\alpha) \cdot f'(\alpha)) D_\alpha(x)$ where the second formula holds due to the fact that $D_\alpha(\beta) = \begin{cases} 1 & \text{if } \beta = \alpha \\ 0 & \text{if } \beta \neq \alpha \end{cases}$.

Lemma 4.1. *For any $\alpha \in (GF(2))^n$, we have (i) $\mu(D_\alpha)(x) = x_1^{a_1} \cdots x_n^{a_n}$ where $\alpha = (a_1, \ldots, a_n)$, (ii) $\mu(x_1^{a_1} \cdots x_n^{a_n}) = D_\alpha(x)$.*

Proof. Due to Theorem 3.6, (i) and (ii) are equivalent. Therefore we only need to prove (ii). Let $\alpha = (a_1, \ldots, a_n)$ be the binary representation of an integer i. It is noted that the truth table of $D_\alpha(x)$ is all-zero vector of length 2^n except for the ith coordinate. By the definition of T_n, the truth table ξ of $x_1^{a_1} \cdots x_n^{a_n}$ is the ith row of T_n. According to Theorem 3.4, $\eta = \xi T_n$ is the truth table

of $\mu(x_1^{a_1} \cdots x_n^{a_n})$. Due to (i) of Lemma 3.3, η is all-zero vector of length 2^n except for the ith coordinate. Therefore $\mu(x_1^{a_1} \cdots x_n^{a_n})$ and $D_\alpha(x)$ have the same truth table and then (ii) holds. □

Due to Formula (2) and Lemma 4.1, we can state as follows.

Theorem 4.2. *Let f be a function on $(GF(2))^n$. Then $\mu(f)(x) = \bigoplus_{\alpha \in (GF(2))^n} f(\alpha) x_1^{a_1} \cdots x_n^{a_n}$.*

5. Computing $\mu(f)$ by Recursive Relations

Theorem 5.1. *As we know, any function f on $(GF(2))^n$ can be expressed as $f(x) = x_1 g(y) \oplus h(y)$ where $x = (x_1, \ldots, x_n)$ and $y = (x_2, \ldots, x_n)$. Then $\mu(f)(x) = x_1(\mu(g)(y) \oplus \mu(h)(y)) \oplus \mu(h)(y)$.*

Proof. Let ξ, η, ζ denote the truth tables of f, g and h respectively. It is easy to verify that $\xi = (\zeta, \eta \oplus \zeta)$. Let ξ' denote the truth table of $\mu(f)$. According to Theorem 3.4, the truth table of $\mu(f)$ can be computed as $\xi T_n = (\zeta, \eta \oplus \zeta) T_n = (\zeta T_{n-1}, \eta T_{n-1})$. Again, due to Theorem 3.4, ζT_{n-1} and ηT_{n-1} are the truth tables of $\mu(h)$ and $\mu(g)$ respectively. Therefore, it is easy to verify that $\mu(f)(x) = x_1(\mu(g)(y) \oplus \mu(h)(y)) \oplus \mu(h)(y)$. □

By Theorem 5.1, we can reduce the size of Möbius Transform.

6. Properties of Möbius Transforms

6.1. Computing $\mu(f)$ after a Permutations on Variables

Notation 4. *Let f be a function on $(GF(2))^n$. Let P be a permutation on $\{1, \ldots, n\}$. Define the function f_P as $f_P(x_1, \ldots, x_n) = f(x_{P(1)}, \ldots, x_{P(n)})$.*

Theorem 6.1. *Let f be a function on $(GF(2))^n$ and $g = \mu(f)$. Then $\mu(f_P) = g_P$ where P is defined in Notation 4.*

Proof. Due to (1), f can be expressed as $f(x_1, \ldots, x_n) = \bigoplus_{\alpha \in (GF(2))^n} g(a_1, \ldots, a_n) x_1^{a_1} \cdots x_n^{a_n}$ where $\alpha = (a_1, \ldots, a_n)$. Then $f_P(x_1, \ldots, x_n) = \bigoplus_{\alpha \in (GF(2))^n} g(a_1, \ldots, a_n) x_{P(1)}^{a_1} \cdots x_{P(n)}^{a_n}$. It is noted that $x_{P(1)}^{a_1} \cdots x_{P(n)}^{a_n}$ is identical with $x_1^{a_{P^{-1}(1)}} \cdots x_n^{a_{P^{-1}(n)}}$ where P^{-1} denotes the inverse of P. Set $a_{P^{-1}(i)} = b_i$ and then $a_i = b_{P(i)}$, $i = 1, \ldots, n$. Therefore $g(a_1, \ldots, a_n) x_{P(1)}^{a_1} \cdots x_{P(n)}^{a_n}$ is identical with $g(b_{P(1)}, \ldots, b_{P(n)}) x_1^{b_1} \cdots x_n^{b_n}$. Then we have proved

that $f_P(x_1,\ldots,x_n) = \bigoplus_{\beta\in(GF(2))^n} g(b_{P(1)},\ldots,b_{P(n)})x_1^{b_1}\cdots x_n^{b_n}$ where $\beta = (b_1,\ldots,b_n)$. By definition, we know that the Möbius transform of f_P is g_P, or in other words, $\mu(f_P) = g_P$. □

We note that the permutation P in Theorem 6.1 defined on $\{1,\ldots,n\}$ cannot be extended to be a permutation on $(GF(2))^n$. This can be seen from the following example. It is easy to verify that $f(x_1,x_2) = x_1 \oplus x_1x_2$ has the Möbius Transform $\mu(f)(x_1,x_2) = x_2$. Set a nonsingular linear transformation on $(GF(2))^2$: $x_1 = y_2$, $x_2 = y_1 \oplus y_2$. It is easy to see that $f(x_1,x_2) = y_1y_2$ whose Möbius Transform is y_1y_2.

6.2. **A Lower Bound on** $deg(f) + deg(\mu(f))$

Theorem 6.2. *Let f be a nonzero function on $(GF(2))^n$. Then $deg(f) + deg(\mu(f)) \geq n$.*

Proof. We prove the theorem by induction on n. It is easy to verify the theorem is true for $n = 1$ because $\mu(f_1) = f_2$, $\mu(f_2) = f_1$ and $\mu(f_3) = f_3$ where $f_1(x_1) = 1 \oplus x_1$, $f_2(x_1) = 1$ and $f_3(x_1) = x_1$. We assume that the theorem holds for $1 \leq n \leq s-1$. Consider the case of $n = s$. Let f be a function on $(GF(2))^s$. We can express f as $f(x) = x_1g(y) \oplus h(y)$ where $x = (x_1,\ldots,x_s)$, $y = (x_2,\ldots,x_s)$, g and h are functions on $(GF(2))^{s-1}$. According to Theorem 5.1, $\mu(f)(x) = x_1(\mu(g)(y) \oplus \mu(h)(y)) \oplus \mu(h)(y)$. There exist two cases to be considered: $g \neq h$ (Case 1) and $g = h$ (Case 2).

We now consider Case 1. Case 1 is composed of three cases: $deg(\mu(g)) > deg(\mu(h))$ (Case 1.1), $deg(\mu(g)) < deg(\mu(h))$ (Case 1.2) and $deg(\mu(g)) = deg(\mu(h))$ (Case 1.3).

For Case 1.1, $deg(f) + deg(\mu(f)) \geq 1 + deg(g) + 1 + deg(\mu(g) \oplus \mu(h)) = 1 + deg(g) + 1 + deg(\mu(g))$. By the induction assumption, $deg(g) + deg(\mu(g)) \geq s-1$ and then $deg(f) + deg(\mu(f)) \geq 1 + s$.

For Case 1.2, $deg(f) + deg(\mu(f)) \geq deg(h) + 1 + deg(\mu(g) \oplus \mu(h)) = deg(h) + 1 + deg(\mu(h))$. By the induction assumption, $deg(h) + deg(\mu(h)) \geq s-1$ and then $deg(f) + deg(\mu(f)) \geq s$.

For Case 1.3, $deg(f) + deg(\mu(f)) \geq 1 + deg(g) + deg(\mu(h)) = 1 + deg(h) + deg(\mu(h))$. By the induction assumption, $deg(h) + deg(\mu(h)) \geq s-1$ and then $deg(f) + deg(\mu(f)) \geq s$.

We next consider Case 2. In Case 2, $deg(f) + deg(\mu(f)) = 1 + deg(g) + deg(\mu(h)) = 1 + deg(h) + deg(\mu(h))$. By the induction assumption, $deg(h) + deg(\mu(h)) \geq s-1$ and then $deg(f) + deg(\mu(f)) \geq$

s. We have proved that the theorem is true for $n = s$. Therefore we have proved the theorem. □

It is noted that the lower bound in Theorem 6.2 can be reached. For example, if $f(x) = (1 \oplus x_1) \cdots (1 \oplus x_n) = D_{\alpha_0}(x)$ where α_0 denotes the zero vector in $(GF(2))^n$, according to Lemma 4.1, $\mu(f)$ is the constant one. Then $deg(f) + deg(\mu(f)) = n + 0 = n$.

7. Concept of Coincident Boolean Functions

In this section we propose a special kind of Boolean functions.

Definition 7.1. Let f be a function on $(GF(2))^n$. If f and $\mu(f)$ are identical, or in other words, $f(\alpha) = 1$ if and only if $x_1^{a_1} \cdots x_n^{a_n}$ is a monomial in the ANF of f, for any $\alpha = (a_1, \ldots, a_n) \in (GF(2))^n$, then f is called a *coincident function*.

By Definition 7.1 and Theorem 3.4, we can conclude

Theorem 7.2. *Let f be a function on $(GF(2))^n$ and $g = \mu(f)$. Denote the truth tables of f and g by ξ and η. Then the following statements are equivalent: (i) f is coincident, (ii) g is coincident, (iii) $\xi T_n = \xi$, (iv) $\eta T_n = \eta$, (v) f and g are identical, (vi) ξ and η identical.*

Example 7.3. Consider the function f on $(GF(2))^4$:
$f(x_1, x_2, x_3, x_4) = x_2x_4 \oplus x_2x_3 \oplus x_1x_2 \oplus x_1x_3x_4 \oplus x_1x_2x_4 \oplus x_1x_2x_3$.
From the ANF of f, we know that the truth table of $\mu(f)$ is (0000011000011110). By computing, the truth table of f is also (0000011000011110). Then f is coincident on $(GF(2))^4$.

Since any coincident function is identical with its Möbius Transform, we can have the truth table/ANF of a coincident function from its ANF/truth table without computing.

8. Characterisations and Constructions of Coincident Functions (by Matrix)

Notation 5. *Set $T_n^* = T_n \oplus I_{2^n}$, $n = 1, 2, \ldots$.*

Due to Theorem 7.2, we can state as follows.

Theorem 8.1. *Let f be a function on $(GF(2))^n$ and $g = \mu(f)$. Then the following statements are equivalent: (i) f is coincident,*

(ii) g is coincident, (iii) the truth table ξ of f satisfies $\xi T_n^* = 0$ where 0 denotes the all-zero vector of length 2^n, (iv) the truth table η of g satisfies $\eta T_n^* = 0$.

Lemma 8.2. *(i)* $T_n^* = \begin{bmatrix} T_{n-1}^* & T_{n-1} \\ O_{2^{n-1}} & T_{n-1}^* \end{bmatrix}$, $n = 1, 2, \ldots$,
(ii) $(T_n^*)^2 = 0_{2^n}$, *(iii)* $T_n T_n^* = T_n^* T_n = T_n^*$.

Proof. (i) is obvious due to the relation between T_n and T_n^*. (ii) and (iii) are equivalent to (ii) and (iii) of Lemma 3.3 respectively. □

Theorem 8.3. *Let f be a function on $(GF(2))^n$. Then the following statements are equivalent: (i) f is coincident, (ii) the truth table of f can be expressed as $(\zeta T_{n-1}^*, \zeta)$ where ζ is a (0, 1)-vector of length 2^{n-1}, (iii) the truth table of f can be expressed as $(\zeta T_{n-1}^*, \zeta \oplus \vartheta T_{n-1}^*)$ where ϑ is a (0, 1)-vector of length 2^{n-1}.*

Proof. Assume that (i) holds. Denote the truth table of f by (ξ_1, ξ_2) where each ξ_i is a (0, 1)-vector of length 2^{n-1}. Due to Theorem 8.1, we know that $(\xi_1, \xi_2) T_n^* = (\xi_1, \xi_2) \begin{bmatrix} T_{n-1}^* & T_{n-1} \\ O_{2^{n-1}} & T_{n-1}^* \end{bmatrix} = 0$ where 0 denotes the all-zero vector of length 2^n. Therefore $\xi_1 T_{n-1}^* = 0$ and $\xi_1 T_{n-1} \oplus \xi_2 T_{n-1}^* = 0$. From the two equations, we know that $\xi_1 (T_{n-1}^* \oplus T_{n-1}) \oplus \xi_2 T_{n-1}^* = 0$ and then $\xi_1 \oplus \xi_2 T_{n-1}^* = 0$ or $\xi_1 = \xi_2 T_{n-1}^*$. Thus the truth table of f can be expressed as $(\xi_2 T_{n-1}^*, \xi_2)$. This proves that (ii) holds. We then have proved (i) $\Longrightarrow$ (ii). Assume that (ii) holds, i.e., the truth table of f can be expressed as $(\zeta T_{n-1}^*, \zeta)$. Let ϑ be any (0, 1)-vector of length 2^{n-1}. Set $\zeta = \zeta' \oplus \vartheta T_{n-1}^*$. Due to Lemma 8.2, $\zeta T_{n-1}^* = \zeta' T_{n-1}^*$. Therefore $(\zeta T_{n-1}^*, \zeta) = (\zeta' T_{n-1}^*, \zeta' \oplus \vartheta T_{n-1}^*)$ and then (iii) holds. We then have proved (ii) $\Longrightarrow$ (iii). Assume that (iii) holds, i.e., the truth table of f can be expressed as $(\zeta T_{n-1}^*, \zeta \oplus \vartheta T_{n-1}^*)$ where both ζ and ϑ are (0, 1)-vector of length 2^{n-1}. By using Lemma 8.2, we know that $(\zeta T_{n-1}^*, \zeta \oplus \vartheta T_{n-1}^*) T_n^* = 0$. Due to Theorem 8.1, f is coincident. This proves (iii) $\Longrightarrow$ (i). □

Theorem 8.4. *Let f be a function on $(GF(2))^n$. Then f is coincident if and only if the truth table of f can be expressed as ηT_n^* where η is a (0, 1)-vector of length 2^n.*

Proof. The sufficiency is true due to Theorem 8.1 and Lemma 8.2. We now prove the necessity. Assume that f is coincident. According to Theorem 8.3, the truth table of f can be expressed

as $(\zeta T^*_{n-1}, \zeta)$ where ζ is a (0, 1)-vector of length 2^{n-1}. By using Lemma 8.2, it is easy to verify that $(\zeta T^*_{n-1}, \zeta) = (\zeta T_{n-1}, 0)T^*_n$. This proves the necessity. □

We can state Theorem 8.4 equivalently.

Theorem 8.5. *Let f be a function on $(GF(2))^n$. Then f is coincident if and only if the truth table of f is a linear combination of rows of T^*_n.*

By the way, we can construct coincident functions by using Theorems 8.3, 8.4 and 8.5.

9. Enumeration of Coincident Functions

Lemma 9.1. *All the 2^{n-1} rows of the matrix $\begin{bmatrix} T^*_{n-1} & T_{n-1} \end{bmatrix}$ form a basis of rows of T^*_n, where $n = 1, 2, \ldots$.*

Proof. Due to Lemma 8.2, $T^*_n = \begin{bmatrix} T^*_{n-1} & T_{n-1} \\ 0_{2^{n-1}} & T^*_{n-1} \end{bmatrix}$. Again, due to Lemma 8.2, $\begin{bmatrix} I_{2^{n-1}} & 0_{2^{n-1}} \\ T^*_{n-1} & I_{2^{n-1}} \end{bmatrix} \begin{bmatrix} T^*_{n-1} & T_{n-1} \\ 0_{2^{n-1}} & 0_{2^{n-1}} \end{bmatrix} = \begin{bmatrix} T^*_{n-1} & T_{n-1} \\ 0_{2^{n-1}} & T^*_{n-1} \end{bmatrix}$. It is noted that T_{n-1} has a rank 2^{n-1}. The proof is completed. □

Due to Theorem 8.5 and Lemma 9.1, we have Theorems 9.2 and 9.3 where Theorem 9.2 is an improvement on Theorem 8.5.

Theorem 9.2. *Let f be a function on $(GF(2))^n$. Then f is coincident if and only if the truth table of f is a linear combination of rows of $\begin{bmatrix} T^*_{n-1} & T_{n-1} \end{bmatrix}$, where $n = 1, 2, \ldots$.*

Theorem 9.3. *There precisely exist $2^{2^{n-1}}$ coincident functions on $(GF(2))^n$ that form a 2^{n-1}-dimensional linear subspace of $\mathcal{R}_n$ where $\mathcal{R}_n$ is defined in Notation 1.*

Example 9.4. According to Theorem 9.3, there precisely exist $2^{2^{3-1}} = 16$ coincident functions on $(GF(2))^3$. According to Theorem 9.2, all the linear combinations of rows of $[T^*_2, T_2] = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ are the truth tables of coincident functions on $(GF(2))^3$: (01111111), (00010101), (00010011), (00000001), (00000111), (00000110), (01101010), (00010100),

(01101101), (01101011), (01111110), (01101100), (01111000), (01111001), (00010010), (00000000). We directly write the ANFs of the 16 coincident functions on $(GF(2))^3$: $x_3 \oplus x_2 \oplus x_1 \oplus x_2x_3 \oplus x_1x_3 \oplus x_1x_2 \oplus x_1x_2x_3$, $x_2x_3 \oplus x_1x_3 \oplus x_1x_2x_3$, $x_2x_3 \oplus x_1x_2 \oplus x_1x_2x_3$, $x_1x_2x_3$, $x_1x_3 \oplus x_1x_2 \oplus x_1x_2x_3$, $x_1x_3 \oplus x_1x_2$, $x_3 \oplus x_2 \oplus x_1 \oplus x_1x_2$, $x_2x_3 \oplus x_1x_3$, $x_3 \oplus x_2 \oplus x_1 \oplus x_1x_3 \oplus x_1x_2x_3$, $x_3 \oplus x_2 \oplus x_1 \oplus x_1x_2 \oplus x_1x_2x_3$, $x_3 \oplus x_2 \oplus x_1 \oplus x_2x_3 \oplus x_1x_3 \oplus x_1x_2$, $x_3 \oplus x_2 \oplus x_1 \oplus x_1x_3$, $x_3 \oplus x_2 \oplus x_1 \oplus x_2x_3$, $x_3 \oplus x_2 \oplus x_1 \oplus x_2x_3 \oplus x_1x_2x_3$, $x_2x_3 \oplus x_1x_2$, 0

10. Characterisations and Constructions of Coincident Functions (by Polynomial)

Definition 10.1. Define a mapping Ψ from $\mathcal{R}_n$ to $\mathcal{R}_n$, where $\mathcal{R}_n$ is defined in Notation 1: $\Psi(f) = h$ if and only if $\xi T_n^* = \zeta$ where $f, h \in \mathcal{R}_n$, ξ and ζ are truth tables of f and h respectively.

By Definition 10.1, Ψ is a linear mapping.

Lemma 10.2. *Let f be a function on $(GF(2))^n$. Then $\Psi(f) = h$ if and only if $f \oplus \mu(f) = h$ where Ψ is defined in Definition 10.1.*

Proof. Let ξ and ζ be the truth tables of f and h respectively. It is clear that $\Psi(f) = h \iff \xi T_n^* = \zeta \iff \xi \oplus \xi T_n = \zeta \iff f \oplus \mu(f) = h$. □

Theorem 10.3. *Let h be a function on $(GF(2))^n$. Then the following statements are equivalent: (i) h is coincident, (ii) there exists a function f on $(GF(2))^n$ such that $h = \Psi(f)$, i.e., $h = f \oplus \mu(f)$, (iii) $\Psi(h)$ is the zero function.*

Proof. Due to Theorem 8.4, (i) $\iff$ (ii). Due to Theorem 8.1, (i) $\iff$ (iii). □

Due to Lemma 4.1 and Theorem 10.3, we can state as follows.

Lemma 10.4. *For any $\alpha = (a_1, \ldots, a_n) \in (GF(2))^n$, $D_\alpha(x) \oplus x_1^{a_1} \cdots x_n^{a_n}$ is coincident.*

Again, by using Theorem 10.3 and Lemma 4.1, we can state more generally.

Theorem 10.5. *Let h be a function on $(GF(2))^n$. Then h is coincident if and only if h is a linear combination of all the functions in the form $D_\alpha(x) \oplus x_1^{a_1} \cdots x_n^{a_n}$ where $\alpha = (a_1, \ldots, a_n) \in (GF(2))^n$.*

Due to Formulas (1), (2) and Definition 7.1, we conclude

Theorem 10.6. *Let f be a function on $(GF(2))^n$ whose ANF of f is given as $f(x) = \bigoplus_{\alpha\in(GF(2))^n} g(\alpha)x_1^{a_1}\cdots x_n^{a_n}$ where $\alpha = (a_1,\ldots,a_n) \in (GF(2))^n$ and $g = \mu(f)$. Then the following statements are equivalent: (i) f is coincident,*
(ii) $f(x) = \bigoplus_{\alpha\in(GF(2))^n} f(\alpha)x_1^{a_1}\cdots x_n^{a_n}$,
(iii) $f(x) = \bigoplus_{\alpha\in(GF(2))^n} g(\alpha)D_\alpha(x)$.

By the way, we can construct coincident functions by using Theorems 10.3, 10.5 and 10.6.

Notation 6. *Let $\beta = (b_1,\ldots,b_n)$ and $\alpha = (a_1,\ldots,a_n)$ be (0, 1)-vectors. Then $\beta \preceq \alpha$ means that if $b_j = 1$ then $a_j = 1$. In particular, $\beta \prec \alpha$ means that $\beta \preceq \alpha$ but $\beta \neq \alpha$.*

The following result can be found in p.372 of [1].

Lemma 10.7. *Let f be a function on $(GF(2))^n$ and $\alpha = (a_1,\ldots,a_n)$ be a vector in $(GF(2))^n$. Then the term $x_1^{a_1}\cdots x_n^{a_n}$ appears in the ANF of f if and only if $\bigoplus_{\beta\preceq\alpha} f(\beta) = 1$.*

Theorem 10.8. *Let f be a function on $(GF(2))^n$. Then f is coincident if and only if for any $\alpha \in (GF(2))^n$, $\bigoplus_{\beta\prec\alpha} f(\beta) = 0$.*

Proof. Let $g = \mu(f)$. Due to Lemma 10.7, $g(\alpha) = \bigoplus_{\beta\preceq\alpha} f(\beta)$ for any $\alpha \in (GF(2))^n$. Then f is coincident $\Longleftrightarrow f = g \Longleftrightarrow$ for each $\alpha \in (GF(2))^n$, $f(\alpha) = \bigoplus_{\beta\preceq\alpha} f(\beta)$ or $\bigoplus_{\beta\prec\alpha} f(\beta) = 0$. □

11. Characterisations and Constructions of Coincident Functions (by Recursive Formulas)

Theorem 11.1. *Let f be a function on $(GF(2))^n$. Then f is coincident if and only if there exists a function g on $(GF(2))^{n-1}$ such that $f(x) = x_1 g(y) \oplus \Psi(g)(y)$ where Ψ has been defined in Definition 10.1. Furthermore, if f is nonzero then g is nonzero.*

Proof. Since f can be expressed as $f(x) = x_1 g(y) \oplus h(y)$ where both g and h are functions on $(GF(2))^{n-1}$, due to Theorem 5.1, $\mu(f)(x) = x_1\mu(g\oplus h)(y)\oplus\mu(h)(y)$. It is noted that f is coincident $\Longleftrightarrow f = \mu(f) \Longleftrightarrow g = \mu(g\oplus h)$ and $h = \mu(h) \Longleftrightarrow h = \mu(h)$ and $h = \mu(g)\oplus g \Longleftrightarrow h = \mu(g)\oplus g$ (due to Theorem 10.3). Due to Lemma 10.2, $g \oplus \mu(g) = \Psi(g)$. This proves the main part of the theorem. Clearly if f is nonzero then g is nonzero. □

Recursively applying Theorem 11.1, we state as follows.

Theorem 11.2. *Let f be a function on $(GF(2))^n$. Then f is coincident if and only if there exists a function f_i on $(GF(2))^{n-i}$, $i = 1, \ldots, n$, such that $f(x_1, \ldots, x_n) = x_1 f_1(x_2, \ldots, x_n) \oplus x_2 f_2(x_3, \ldots, x_n) \oplus \cdots \oplus x_{n-1} f_{n-1}(x_n) \oplus f_n(x_n)$ where $x_i f_i(x_{i+1}, \ldots, x_n) \oplus \cdots \oplus x_{n-1} f_{n-1}(x_n) \oplus f_n(x_n) = \Psi(x_{i-1} f_{i-1}(x_i, \ldots, x_n) \oplus \cdots \oplus x_{n-1} f_{n-1}(x_n) \oplus f_n(x_n))$, $i = 2, \ldots, n$.*

By the way, we can construct coincident functions by using Theorems 11.1 and 11.2.

12. Properties of Coincident Functions

12.1. General Properties

Theorem 12.1. *Let f be a function on $(GF(2))^n$ and P be a permutation on $\{1, \ldots, n\}$. Then f is coincident if and only if f_P is coincident, where f_P is defined in Notation 4, i.e., $f_P(x_1, \ldots, x_n) = f(x_{P(1)}, \ldots, x_{P(n)})$.*

Proof. Set $g = \mu(f)$. Assume that f is coincident. Then g is identical with f and then $f_P = g_P$. By Theorem 6.1, $\mu(f_P) = g_P$. Then we have $\mu(f_P) = f_P$ and then f_P is coincident. The converse is true because we can set $f_P = f'$ then $f'_{P^{-1}} = f$. □

We note that the permutation P in Theorem 12.1 defined on $\{1, \ldots, n\}$ cannot be extended to be a permutation on $(GF(2))^n$. This can be seen from the following example. From Example 9.4, $f(x_1, x_2, x_3) = x_1 x_2 x_3$ is coincident on $(GF(2))^3$. Set a nonsingular linear transformation on $(GF(2))^3$: $x_1 = y_1 \oplus y_2$, $x_2 = y_2$, $x_3 = y_3$. It is easy to see that $f(x_1, x_2, x_3) = y_1 y_2 y_3 \oplus y_2 y_3$ is not coincident on $(GF(2))^3$.

Theorem 12.2. *Let f be a function on $(GF(2))^n$ and P be a permutation on $\{1, \ldots, n\}$. Set $f'(x_{P(1)}, \ldots, x_{P(n)}) = f(x_1, \ldots, x_n)$. Then f is coincident if and only if f' is coincident.*

Proof. The theorem is true due to the equivalence between (i) and (iii) in Theorem 10.6. □

A difference between Theorems 12.1 and 12.2 is that the permutation P in Theorem 12.1 replaces x_j by $x_{P(j)}$ while P in Theorem 12.2 regards $x_{P(j)}$ as the jth variable but does not change the function f. For example, if $f(x_1, x_2, x_3) = x_1 x_2 \oplus x_2 x_3$, $P(1) = 2$, $P(2) = 3$ and $P(3) = 1$, then $f_P(x_1, x_2, x_3) = x_2 x_3 \oplus x_3 x_1$ but

$f'(x_2, x_3, x_1) = x_1x_2 \oplus x_2x_3$ where f' is mentioned in Theorem 12.2.

Theorem 12.3. *Let f be a nonzero coincident function on $(GF(2))^n$. Then each variable x_j appears in a monomial of the ANF of f.*

Proof. By Theorem 11.1, $f(x) = x_1g(y) \oplus \Psi(g)(y)$ where g is a function on $(GF(2))^{n-1}$. Since f is nonzero, g is nonzero. Then x_1 appears in a monomial of the ANF of f. Therefore, if we regard any other variable x_j as the 1st variable, By Theorem 12.2, the new function f' is also coincident. By the same reasoning, x_j appears in a monomial of the ANF of f' as well as x_1 in f. □

Corollary 12.4. *Let f be function on $(GF(2))^n$ and $g = \mu(f)$. If f is a coincident function then $f(0) = 0$ and $g(0) = 0$.*

Proof. Due to Theorem 8.4, the truth table of f can be expressed as ξT_n^*. It is noted that the leftmost column of T_n^* is the all-zero column. Then the first coordinate of ξT_n^* turns out to be zero. This proves that $f(0) = 0$ and then $g(0) = 0$. □

Theorem 12.5. *Let f be a coincident function on $(GF(2))^n$. Then either the ANF of f has every linear term x_j, or, the ANF does not have any linear term.*

Proof. Assume that the ANF of f has a linear term x_{j_0} where $1 \leq j_0 \leq n$. Let $i_0 \in \{1, \ldots, n\} - \{j_0\}$. Without loss of generality, we assume that $i_0 < j_0$. Let γ_i denote the vector in $(GF(2))^n$ whose ith coordinate is one and all other coordinates are zero. Let $\gamma_{i,j}$ denote the vector in $(GF(2))^n$ whose ith and jth coordinates are one and all other coordinates are zero. According to Theorem 10.8, $\bigoplus_{\beta \prec \gamma_{i_0,j_0}} g(\beta) = 0$. More precisely, $g(0) \oplus g(\gamma_{j_0}) \oplus g(\gamma_{i_0}) = 0$. Due to Corollary 12.4, $g(0) = 0$. Since the ANF of f has a linear term x_{j_0}, $g(\gamma_{j_0}) = 1$. Therefore we know that $g(\gamma_{i_0}) = 1$. This means that the ANF of f has a linear term x_{i_0}. Since i_0 is arbitrarily included in $\{1, \ldots, n\} - \{j_0\}$, we have proved the theorem. □

Lemma 12.6. *Let f be a coincident function on $(GF(2))^n$. Then for any integer r with $1 \leq r \leq n-1$ and the r-subset $\{1, \ldots, r\}$ of $\{1, \ldots, n\}$, $f(x_1, \ldots, x_n)|_{x_1=0,\ldots,x_r=0}$ is a coincident function on $(GF(2))^{n-r}$.*

Proof. According to Theorem 11.1, $f(x) = x_1g(y) \oplus \Psi(g)(y)$ where Ψ has been defined in Definition 10.1. Then $f(0, x_2, \ldots, x_n) =$

$\Psi(g)(x_2, \ldots, x_n)$. Due to Theorem 10.3, $\Psi(g)$ is a coincident function on $(GF(2))^{n-1}$, i.e., $f(0, x_2, \ldots, x_n)$ is a coincident function on $(GF(2))^{n-1}$. Applying the same reasoning to $\Psi(g)$, we know that $f(0, 0, x_3, \ldots, x_n)$ is a coincident function on $(GF(2))^{n-2}$. Repeatedly, we can prove that $f(0, \ldots, 0, x_{r+1}, \ldots, x_n)$ is a coincident function on $(GF(2))^{n-r}$. □

Theorem 12.7. *Let f be a coincident function on $(GF(2))^n$. Then for any integer r with $1 \leq r \leq n-1$ and any r-subset $\{j_1, \ldots, j_r\}$ of $\{1, \ldots, n\}$,*
$f(x_1, \ldots, x_n)|_{x_{j_1}=0,\ldots,x_{j_r}=0}$ is a coincident function on $(GF(2))^{n-r}$.

Proof. Let $\{j_1, \ldots, j_r\} \cup \{j_{r+1}, \ldots, j_n\} = \{1, \ldots, n\}$. We define a function f': $f'(x_{j_1}, \ldots, x_{j_n}) = f(x_1, \ldots, x_n)$. According to Theorem 12.2, f' is coincident. Applying Lemma 12.6 to f', we have proved the theorem. □

12.2. **A Lower Bound on Degree of Coincident Functions**

Lemma 12.8. *There precisely exist $2^{2^{n-1}-1}$ coincident functions on $(GF(2))^n$ having a degree n and there precisely exist $2^{2^{n-1}-1}$ coincident functions on $(GF(2))^n$ having a degree less than n.*

Proof. Due to Theorem 8.5, the truth table of a coincident function f on $(GF(2))^n$ is a linear combination of rows of T_n^*. It is noted that the rightmost column of T_n^* contains ones. Then there precisely 50% such linear combinations whose last coordinate is one. By Definition 7.1, for any coincident function f on $(GF(2))^n$, $deg(f) = n$ if and only if $f(1, \ldots, 1) = 1$. Therefore there precisely 50% coincident functions on $(GF(2))^n$ having a degree n. Therefore, due to Theorem 9.3, we have proved the corollary. □

We next indicate that all coincident functions have a high degree even for coincident functions whose degree are less than n.

Theorem 12.9. *Let f be a coincident function on $(GF(2))^n$. Then $deg(f) \geq \lceil \frac{1}{2}n \rceil$. More precisely, (i) $deg(f) \geq \frac{1}{2}n$ where n is even, (ii) $deg(f) \geq \frac{1}{2}(n+1)$ where n is odd.*

Proof. According to Theorem 6.2, $deg(f) + deg(\mu(f)) \geq n$. On the other hand, since f is coincident, f and $\mu(f)$ are identical. Then $2deg(f) \geq n$ and then $deg(f) \geq \frac{1}{2}n$. In particular, when n is odd, it is noted that $deg(f) \geq \frac{1}{2}n$. Since n is odd and $deg(f)$ is integer, $deg(f) \geq \frac{1}{2}(n+1)$. Summarily, $deg(f) \geq \lceil \frac{1}{2}n \rceil$. □

We now indicate that the lower bound in Theorem 12.9 is tight. For example, $f(x_1, x_2, x_3, x_4) = x_2x_4 \oplus x_2x_3 \oplus x_1x_4 \oplus x_1x_3$ is a coincident function on $(GF(2))^4$ having a degree two. $f(x_1, x_2, x_3) = x_3 \oplus x_2 \oplus x_1 \oplus x_2x_3 \oplus x_1x_3 \oplus x_1x_2$ is a coincident function on $(GF(2))^3$ having a degree two.

13. Coincident Functions with High Nonlinearity and High Degree

The *nonlinearity* N_f of a function f on $(GF(2))^n$ is defined as $N_f = \min_{i=1,2,\ldots,2^{n+1}} d(f, \psi_i)$ where ψ_1, ψ_2, ..., $\psi_{2^{n+1}}$ are all the affine functions on $(GF(2))^n$. It is well-known that for any function f on $(GF(n))^n$, the nonlinearity N_f of f satisfies $N_f \leq 2^{n-1} - 2^{\frac{1}{2}n-1}$. We can define bent functions, introduced first by Rothaus [2], equivalently as follows: a function f on $(GF(n))^n$ is said to be *bent* if the nonlinearity N_f reaches the maximum value $N_f = 2^{n-1} - 2^{\frac{1}{2}n-1}$. Obviously bent functions on $(GF(2))^n$ exist for even n.

13.1. Construction 1 (for Case of Even Variables)

The following statement can be verified straightforwardly.

Lemma 13.1. *Let f_1, f_2 and f_3 be functions on $(GF(2))^n$. Then $d(f_1, f_3) \leq d(f_1, f_2) + d(f_2, f_3)$.*

Theorem 13.2. *Let $f(x_1, \ldots, x_{2k}) = x_1x_2 \oplus \cdots \oplus x_{2k-1}x_{2k}$. Set $h = f \oplus \mu(f)$. Then (i) h is a coincident function on $(GF(2))^{2k}$, (ii) $N_h \geq 2^{2k-1} - 2^{k-1} - k$, (iii) $deg(h) \geq 2k - 2$.*

Proof. Due to Theorem 10.3, h is coincident. Let ξ and η be the truth tables of f and $\mu(f)$ respectively. Then $\xi \oplus \eta$ is the truth table of h. Let ψ be an affine function on $(GF(2))^{2k}$ and ℓ be the truth table of ψ. By the definition of nonlinearity, $d(\xi, \ell) \geq N_f$. On the other hand, it is obvious that $HW(\eta) = k$. Therefore $d(\xi \oplus \eta, \xi) = k$. Due to Lemma 13.1, $d(\xi, \ell) \leq d(\xi, \xi \oplus \eta) + d(\xi \oplus \eta, \ell)$. Then $N_f \leq k + d(\xi \oplus \eta, \ell)$ or $d(\xi \oplus \eta, \ell) \geq N_f - k$. Since ψ is an arbitrarily affine function, $N_h \geq N_f - k$. It is well-known that f is bent. Then $N_f = 2^{2k-1} - 2^{k-1}$. We then have proved that $N_h \geq 2^{2k-1} - 2^{k-1} - k$. Due to Theorem 6.2, $deg(f) \oplus deg(\mu(f)) \geq 2k$. Since $deg(f) = 2$, we know that $deg(\mu(f)) \geq 2k - 2$. Clearly $deg(h) = deg(\mu(f))$, We have proved the theorem. □

13.2. **Construction 2 (for Case of Odd Variables)**

Theorem 13.3. *Let* $f(x_1, x_2, \ldots, x_{2k+1}) = x_2x_3 \oplus x_4x_5 \cdots \oplus x_{2k}x_{2k+1}$. *Set* $h = f \oplus \mu(f)$. *Then (i)* h *is a coincident function on* $(GF(2))^{2k+1}$, *(ii)* $N_h \geq 2^{2k} - 2^k - k$, *(iii)* $deg(h) \geq 2k - 1$.

Proof. By using the same reasoning in the proof of Theorem 13.2, we have $N_h \geq N_f - k$. Set $f'(x_2, \ldots, x_{2k+1}) = x_2x_3 \oplus x_4x_5 \cdots \oplus x_{2k}x_{2k+1}$. Then f' is a bent function on $(GF(2))^{2k}$ and then $N_{f'} = 2^{2k-1} - 2^{k-1}$. It is easy to see that $N_f = 2N_{f'} = 2^{2k} - 2^k$. Therefore we have proved that $N_h \geq 2^{2k} - 2^k - k$. Due to Theorem 6.2, $deg(f) \oplus deg(\mu(f)) \geq 2k + 1$. Since $deg(f) = 2$, we know that $deg(\mu(f)) \geq 2k + 1 - 2 = 2k - 1$. Clearly $deg(h) = deg(\mu(f))$, We have proved the theorem. □

Both coincident functions in Theorems 13.2 and 13.3 are highly nonlinear, comparing to the maximum nonlinearity $2^{n-1} - 2^{\frac{1}{2}n-1}$ of functions on $(GF(2))^n$. Obviously the two coincident functions are also of high algebraic degree.

14. **Conclusions**

We have established relations between Boolean functions and their Möbius transforms so as to compute the truth table/ANF from the ANF/truth table of a function in different conditions and find new properties of Boolean functions. We have proposed a special kind of Boolean functions, so-called coincident functions, and extensively studied such functions.

References

[1] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, New York, Oxford, 1978.

[2] O. S. Rothaus. On "bent" functions. *Journal of Combinatorial Theory*, Ser. A, 20:300–305, 1976.

[3] Y. Zheng, X. M. Zhang, and Hideki Imai. Restrictions, terms and nonlinearity of boolean functions. *Theoretical Computer Science*, 226:207–223, 1999.

BFCA'07 - Proceedings

About the book
Held during May, 2007, in Paris, BFCA'07 was the third workshop about Boolean Functions and their applications (notably in cryptography). During two days, many different international scientists met each other and talked about their work. This book contains the acts of the proceedings of the conference.

À propos de cet ouvrage
En Mai 2007 s'est tenu à Paris, BFCA'07, le troisième atelier sur le thème des Fonctions Booléennes et de leurs applications (en particulier cryptographiques). Pendant deux jours, de nombreux chercheurs internationaux s'y sont rencontrés et y ont parlé de leurs travaux. Cet ouvrage est composé des articles associés aux différentes conférences qui s'y sont tenues.

About the editors :
Jean-Francis Michon is Computer Science Professor and Director of the LITIS (Computer Science Laboratory) at University of Rouen, France.
Pierre Valarcher is Computer Science Assistant Professor and Member of the LACL (Computer Science Laboratory) at University of Paris-East, France.
Jean-Baptiste Yunès is Computer Science Assistant Professor and Member of the LIAFA (Computer Science Laboratory) at University of Paris 7 - Denis Diderot, France.

With the collaboration of :
Frederik Armknecht, Pierre-Louis Cayrel, Joan-Josep Climent, Deepak Kumar Dalai, José Raimundo de Oliveira, Maurício Araújo Dias, Ali Doğanaksoy, Philippe Gaborit, Francisco J. García, Sylvain Guilley, Philippe Hoogvorst, Subhamoy Maitra, Renaud Pacalet, Josef Pieprzyk, Verónica Requena, Olivier Ruatta, Sumanta Sarkar, Elif Saygı, Zülfükar Saygı, Johannes Schmidt, İsa Sertkaya, Xian-Mo Zhang