



**HAL**  
open science

## How to Take a Function Apart with SboxU

Léo Perrin

► **To cite this version:**

Léo Perrin. How to Take a Function Apart with SboxU. BFA 2020 - The 5th International Workshop on Boolean Functions and their Applications, Sep 2020, Loen, Norway. hal-03136551

**HAL Id: hal-03136551**

**<https://inria.hal.science/hal-03136551v1>**

Submitted on 9 Feb 2021

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

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

# How to Take a Function Apart with SboxU

(Also Featuring some New Results on Ortho-Derivatives)

Anne Canteaut<sup>1</sup>, Léo Perrin<sup>1</sup>

<sup>1</sup>Inria, France

leo.perrin@inria.fr

 @lpp\_crypto



Boolean Functions and their Applications 2020





**A wild vectorial Boolean function appears!**



A wild vectorial Boolean function appears!

What do you do?

# Outline

- 1 Basic Functionalities
- 2 CCZ-Equivalence
- 3 Ortho-Derivative
- 4 Conclusion

## Plan of this Section

- 1 Basic Functionalities**
  - Installation
  - Core Functionalities
- 2 CCZ-Equivalence
- 3 Ortho-Derivative
- 4 Conclusion

## Plan of this Section

- 1 **Basic Functionalities**
  - Installation
  - Core Functionalities
- 2 CCZ-Equivalence
- 3 Ortho-Derivative
- 4 Conclusion



## How to

- You need to have SAGE installed
- Then head to `https://github.com/lpp-crypto/sboxU`

Demo

## Sbox from SAGE vs. sboxU

There are already many functions for investigating vectorial boolean functions in SAGE:

- Class `SBox` from `sage.crypto.sbox` (or `sage.crypto.mq.sbox` in older versions)
- Module `boolean_function` from `sage.crypto`

## Sbox from SAGE vs. sboxU

There are already many functions for investigating vectorial boolean functions in SAGE:

- Class `SBox` from `sage.crypto.sbox` (or `sage.crypto.mq.sbox` in older versions)
- Module `boolean_function` from `sage.crypto`

### SAGE `SBox`

- Supports output size  $\neq$  input size
- Sub-routines written in Python or Cython
- Built-in SAGE

### `sboxU`

- **Assumes** output size = input size
- Sub-routines written in Python or multi-threaded C++
- Cutting edge functionalities

## Plan of this Section

- 1 Basic Functionalities**
  - Installation
  - **Core Functionalities**
- 2 CCZ-Equivalence
- 3 Ortho-Derivative
- 4 Conclusion

## Some Tools

### 1 DDT/LAT (+ Pollock representation thereof)

Demo

## Some Tools

- 1 DDT/LAT (+ Pollock representation thereof)

Demo

- 2 ANF, algebraic degree

Demo

## Some Tools

- 1 DDT/LAT (+ Pollock representation thereof)

Demo

- 2 ANF, algebraic degree

Demo

- 3 Finite field arithmetic

Demo

## Some Tools

1 DDT/LAT (+ Pollock representation thereof)

Demo

2 ANF, algebraic degree

Demo

3 Finite field arithmetic

Demo

4 Linear mappings

Demo



## Plan of this Section

- 1 Basic Functionalities
- 2 **CCZ-Equivalence**
  - Definition and Basic Theorems
  - How Can sboxU Help?
- 3 Ortho-Derivative
- 4 Conclusion

## Plan of this Section

- 1 Basic Functionalities
- 2 **CCZ-Equivalence**
  - **Definition and Basic Theorems**
  - How Can sboxU Help?
- 3 Ortho-Derivative
- 4 Conclusion

## CCZ- and EA-equivalence

### Definition (CCZ-Equivalence)

$F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$  and  $G : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$  are *C(arlet)-C(harpin)-Z(inoviev)* equivalent if

$$\Gamma_G = \{(x, G(x)), \forall x \in \mathbb{F}_2^n\} = L(\{(x, F(x)), \forall x \in \mathbb{F}_2^n\}) = L(\Gamma_F),$$

where  $L : \mathbb{F}_2^{n+m} \rightarrow \mathbb{F}_2^{n+m}$  is an affine permutation.

## CCZ- and EA-equivalence

### Definition (CCZ-Equivalence)

$F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$  and  $G : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$  are *C(arlet)-C(harpin)-Z(inoviev)* equivalent if

$$\Gamma_G = \{(x, G(x)), \forall x \in \mathbb{F}_2^n\} = L(\{(x, F(x)), \forall x \in \mathbb{F}_2^n\}) = L(\Gamma_F),$$

where  $L : \mathbb{F}_2^{n+m} \rightarrow \mathbb{F}_2^{n+m}$  is an affine permutation.

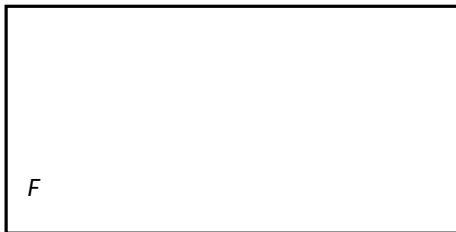
### Definition (EA-Equivalence; EA-mapping)

$F$  and  $G$  are *E(xtended) A(ffine) equivalent* if  $G(x) = (B \circ F \circ A)(x) + C(x)$ , where  $A, B, C$  are affine and  $A, B$  are permutations; so that

$$\{(x, G(x)), \forall x \in \mathbb{F}_2^n\} = \begin{bmatrix} A^{-1} & 0 \\ CA^{-1} & B \end{bmatrix} (\{(x, F(x)), \forall x \in \mathbb{F}_2^n\}).$$

## Some Algorithmic Problems with CCZ-Equivalence

CCZ-class



# Some Algorithmic Problems with CCZ-Equivalence

CCZ-class

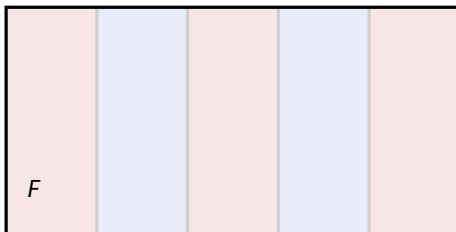
EA-class

EA-class

EA-class

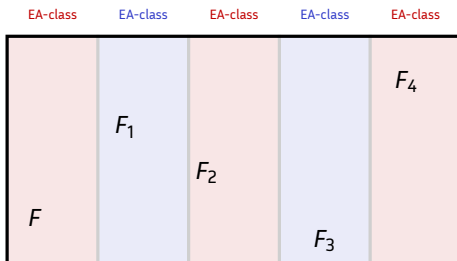
EA-class

EA-class



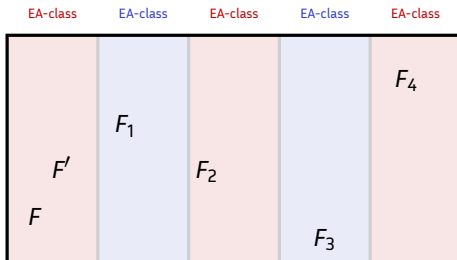
## Some Algorithmic Problems with CCZ-Equivalence

CCZ-class



## Some Algorithmic Problems with CCZ-Equivalence

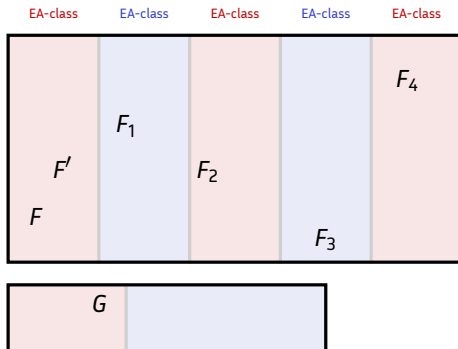
### CCZ-class





# Some Algorithmic Problems with CCZ-Equivalence

## CCZ-class



## Plan of this Section

- 1 Basic Functionalities
- 2 **CCZ-Equivalence**
  - Definition and Basic Theorems
  - **How Can sboxU Help?**
- 3 Ortho-Derivative
- 4 Conclusion

## Exploring a CCZ-class

Demo

## Class Invariants

### Definition (Differential spectrum)

Recall that  $DDT_F[a, b] = \#\{x, F(x + a) + F(x) = b\}$ . The **differential spectrum** is the number of occurrences of each number in the DDT.

## Class Invariants

### Definition (Differential spectrum)

Recall that  $DDT_F[a, b] = \#\{x, F(x + a) + F(x) = b\}$ . The **differential spectrum** is the number of occurrences of each number in the DDT.

### Definition (Walsh spectrum)

Recall that  $\mathcal{W}_F[a, b] = \sum_x (-1)^{a \cdot x + b \cdot F(x)}$ . The **Walsh spectrum** is the number of occurrences of each number in the LAT. The **extended Walsh spectrum** considers only absolute values.

## Class Invariants

### Definition (Differential spectrum)

Recall that  $DDT_F[a, b] = \#\{x, F(x + a) + F(x) = b\}$ . The **differential spectrum** is the number of occurrences of each number in the DDT.

### Definition (Walsh spectrum)

Recall that  $\mathcal{W}_F[a, b] = \sum_x (-1)^{a \cdot x + b \cdot F(x)}$ . The **Walsh spectrum** is the number of occurrences of each number in the LAT. The **extended Walsh spectrum** considers only absolute values.

- Differential and extended Walsh spectra are constant in a **CCZ**-class.
- The algebraic degree and the **thickness spectrum** are constant in an **EA**-class.

Demo

## Plan of this Section

- 1 Basic Functionalities
- 2 CCZ-Equivalence
- 3 Ortho-Derivative**
  - Definition and Basic Theorems
  - Algorithmic Uses
  - Inverting the DDT of a Quadratic Function
- 4 Conclusion

## Plan of this Section

- 1 Basic Functionalities
- 2 CCZ-Equivalence
- 3 Ortho-Derivative**
  - Definition and Basic Theorems
  - Algorithmic Uses
  - Inverting the DDT of a Quadratic Function
- 4 Conclusion



## Definition

### Definition

Ortho-Derivative Let  $F$  be a quadratic function of  $\mathbb{F}_2^n$ . The **ortho-derivatives** of  $F$  are the functions of  $\mathbb{F}_2^n$  such that

$$\forall x \in \mathbb{F}_2^n, \pi_F(a) \cdot \underbrace{(F(x+a) + F(x) + F(a) + F(0))}_{\Delta_a F(x)} = 0.$$

## Definition

### Definition

Ortho-Derivative Let  $F$  be a quadratic function of  $\mathbb{F}_2^n$ . The **ortho-derivatives** of  $F$  are the functions of  $\mathbb{F}_2^n$  such that

$$\forall x \in \mathbb{F}_2^n, \pi_F(a) \cdot \left( \underbrace{F(x+a) + F(x)}_{\Delta_a F(x)} + F(a) + F(0) \right) = 0.$$

- $\pi_F(a)$  is orthogonal to the linear part of the hyperplane  $\text{Im}(\Delta_a F)$
- $\pi_F$  can take any value in  $\mathbb{F}_2$ .

## Basic Properties

### Lemma (Ortho-derivatives of APN functions)

$F$  is APN if and only if  $\pi_F(a)$  is uniquely defined for all  $a \in (\mathbb{F}_2^n)^*$ .

---

<sup>1</sup>See also *A note on the properties of associated Boolean functions of quadratic APN functions* by Anastasiya Gorodilova on ArXiv.

## Basic Properties

### Lemma (Ortho-derivatives of APN functions)

$F$  is APN if and only if  $\pi_F(a)$  is uniquely defined for all  $a \in (\mathbb{F}_2^n)^*$ .

### Lemma (Interaction with EA-equivalence)

If  $G = B \circ F \circ A$  where  $A$  and  $B$  are linear permutations, then

$$\pi_F = (B^T)^{-1} \circ F \circ A$$

---

<sup>1</sup>See also *A note on the properties of associated Boolean functions of quadratic APN functions* by Anastasiya Gorodilova on ArXiv.

## Basic Properties

### Lemma (Ortho-derivatives of APN functions)

$F$  is APN if and only if  $\pi_F(a)$  is uniquely defined for all  $a \in (\mathbb{F}_2^n)^*$ .

### Lemma (Interaction with EA-equivalence)

If  $G = B \circ F \circ A$  where  $A$  and  $B$  are linear permutations, then

$$\pi_G = (B^T)^{-1} \circ \pi_F \circ A$$

It seems like<sup>1</sup> the algebraic degree of the ortho-derivative of an APN function is **always**  $n - 2$ .

---

<sup>1</sup>See also *A note on the properties of associated Boolean functions of quadratic APN functions* by Anastasiya Gorodilova on ArXiv.

## Preimages of the Ortho-Derivative

### Theorem

*Linear Structures (APN case) If*

$$T_F(b) = \{x \in \mathbb{F}_2^n : \pi_F(x) = b\},$$

*then  $T_F(b) = \text{LS}(x \mapsto b \cdot F(x))$ .*

### Corollary

*For any  $b$ ,  $T_F(b)$  is a linear subspace of  $\mathbb{F}_2^n$  whose dimension has the same parity as  $n$ . Furthermore,*

$$(\mathcal{W}_F[a, b])^2 \in \{0, 2^{n+\dim T_F(b)}\}$$

## Plan of this Section

- 1 Basic Functionalities
- 2 CCZ-Equivalence
- 3 Ortho-Derivative**
  - Definition and Basic Theorems
  - Algorithmic Uses**
  - Inverting the DDT of a Quadratic Function
- 4 Conclusion

## Identifying EA- and CCZ-classes

### Corollary (Ortho-derivatives of APN functions)

*The **differential** and **extended Walsh spectra** of the ortho-derivative of an APN function is the same within an EA-class.*



## Identifying EA- and CCZ-classes

### Corollary (Ortho-derivatives of APN functions)

*The **differential** and **extended Walsh spectra** of the ortho-derivative of an APN function is the same within an EA-class.*

### Observation

In practice, these spectra differ from one EA-class to the next!

## Identifying EA- and CCZ-classes

### Corollary (Ortho-derivatives of APN functions)

*The **differential** and **extended Walsh spectra** of the ortho-derivative of an APN function is the same within an EA-class.*

### Observation

In practice, these spectra differ from one EA-class to the next!

We can use this to very efficiently sort large numbers of quadratic functions into distinct EA-classes.

Demo

## Plan of this Section

- 1 Basic Functionalities
- 2 CCZ-Equivalence
- 3 Ortho-Derivative**
  - Definition and Basic Theorems
  - Algorithmic Uses
  - Inverting the DDT of a Quadratic Function**
- 4 Conclusion

## Principle

Is it possible to recover  $F$  given  $\pi_F$ ?

## Principle

Is it possible to recover  $F$  given  $\pi_F$ ? **Yes!**

### The Key Observation

We can write the scalar product  $x \cdot y$  as  $(\vec{x})^T \times \vec{y}$ , where  $\times$  is a matrix operation.

# Principle

Is it possible to recover  $F$  given  $\pi_F$ ? **Yes!**

## The Key Observation

We can write the scalar product  $x \cdot y$  as  $(\vec{x})^T \times \vec{y}$ , where  $\times$  is a matrix operation.

We represent  $F$  as a vector of  $\mathbb{F}_2^{n2^n}$  by concatenating the  $n$ -bit representation of each of the  $2^n$  values  $F(x)$ :

$$\text{vec}(F) = \begin{bmatrix} F_0(0) \\ F_1(0) \\ \dots \\ F_{n-1}(0) \\ F_0(1) \\ \dots \\ F_{n-1}(2^n - 1) \end{bmatrix}.$$

## Re-Defining Ortho-Derivatives

Let  $G$  be a function and  $\zeta_a(G)$  be a matrix defined by

$$\begin{aligned} \mathbf{1} \quad \zeta_G(a)[x, x] &= G(\vec{a})^T, & \zeta_G(a)[x, x + a] &= G(\vec{a})^T, \\ \mathbf{2} \quad \zeta_G(a)[x, 0] &= G(\vec{a})^T, & \zeta_G(a)[x, a] &= G(\vec{a})^T, \end{aligned}$$

## Re-Defining Ortho-Derivatives

Let  $G$  be a function and  $\zeta_a(G)$  be a matrix defined by

$$\begin{aligned} \mathbf{1} \quad \zeta_G(a)[x, x] &= G(\vec{a})^T, & \zeta_G(a)[x, x+a] &= G(\vec{a})^T, \\ \mathbf{2} \quad \zeta_G(a)[x, 0] &= G(\vec{a})^T, & \zeta_G(a)[x, a] &= G(\vec{a})^T, \end{aligned}$$

so that

$$\zeta_G(a) \times \text{vec}(F) = \begin{bmatrix} G(a) \cdot (F(0) + F(0+a) + F(a) + F(0)) \\ G(a) \cdot (F(1) + F(1+a) + F(a) + F(1)) \\ \dots \\ G(a) \cdot (F(2^n-1) + F(2^n-1+a) + F(a) + F(2^n-1)) \end{bmatrix},$$



## Re-Defining Ortho-Derivatives

Let  $G$  be a function and  $\zeta_a(G)$  be a matrix defined by

$$\begin{aligned} \mathbf{1} \quad \zeta_G(a)[x, x] &= G(\vec{a})^T, & \zeta_G(a)[x, x+a] &= G(\vec{a})^T, \\ \mathbf{2} \quad \zeta_G(a)[x, 0] &= G(\vec{a})^T, & \zeta_G(a)[x, a] &= G(\vec{a})^T, \end{aligned}$$

so that

$$\zeta_G(a) \times \text{vec}(F) = \begin{bmatrix} G(a) \cdot (F(0) + F(0+a) + F(a) + F(0)) \\ G(a) \cdot (F(1) + F(1+a) + F(a) + F(1)) \\ \dots \\ G(a) \cdot (F(2^n-1) + F(2^n-1+a) + F(a) + F(2^n-1)) \end{bmatrix},$$

from which we deduce that if  $\pi_F$  is an ortho-derivative of  $F$  then

$$\text{vec}(F) \in \ker(\zeta(\pi_F)) \quad \text{where} \quad \zeta(\pi_F) = \begin{bmatrix} \zeta_0(\pi_F) \\ \dots \\ \zeta_{2^n-1}(\pi_F) \end{bmatrix}.$$

## Inverting the DDT of a Quadratic Function

- 1 Find a DDT,
- 2 deduce the corresponding  $\pi$ ,
- 3 build  $\zeta(\pi)$ ,
- 4 find  $\ker(\zeta(\pi))$ ,
- 5 obtain  $\text{vec}(F)$ !

---

<sup>2</sup>Tricks are used to get rid of redundancies in  $\zeta$ , and trivial solutions.

## Inverting the DDT of a Quadratic Function

- 1 Find a DDT,
- 2 deduce the corresponding  $\pi$ ,
- 3 build  $\zeta(\pi)$ ,
- 4 find  $\ker(\zeta(\pi))$ ,
- 5 obtain  $\text{vec}(F)$ !

In practice, starting from “cleverly” built functions  $\pi$  yields  $\zeta(\pi)$  with empty<sup>2</sup> kernels...

---

<sup>2</sup>Tricks are used to get rid of redundancies in  $\zeta$ , and trivial solutions.

## Plan of this Section

- 1 Basic Functionalities
- 2 CCZ-Equivalence
- 3 Ortho-Derivative
- 4 Conclusion**

## Conclusion

Go an use `sboxU`!

## Conclusion

Go an use sbxU!

**Thank you!**