Bachelor thesis
Computing Science


Radboud University

# Differential Analysis of SATURNIN and SPONGENT

*Author:*
Giovanni Uchoa de Assis
s1030506

*First supervisor/assessor:*
prof. dr. J.J.C. Daemen (Joan)
j.daemen@cs.ru.nl

*Second supervisor:*
ir. D.W.C. Kuijsters (Daniël)
d.kuijsters@cs.ru.nl

*Second assessor:*
dr. ir. B.J.M. Mennink (Bart)
b.mennink@cs.ru.nl

November 3, 2021

**Abstract**

There are many different methods for analyzing block ciphers and cryptographic permutations. The most common method is to study how the round functions that make up each cipher or permutation work in isolation, e.g., looking at the branch number of mixing layers or the nonlinearity of S-boxes. However, a less common approach is to study how the different functions within the permutation interact with each other. One way to do this consists of investigating the linear and differential propagation properties. Within this thesis, permutations obtained by fixing the key in the block cipher SATURNIN to 0 and another one used by the construction SPONGENT will be analyzed by looking at their differential propagation. We will show how their differential trails cluster in differentials. We will also show how differential and differential trail weights clip due to there being an upper bound on these possible weights.

**Keywords:** symmetric cryptography, permutations, round functions, clustering, differential cryptanalysis, clipping

# Contents

## Foreword

We would like to thank Joan Daemen, Daniel Kuijsters and Denise Verbakel for helping out and giving advice on this paper.

# Chapter 1

# Introduction

Encryption is used everywhere in the modern world. One popular method of encryption consists of using block ciphers and permutations as building blocks for encryption.
Different ciphers or permutations are designed with different goals in mind. One type of cipher that is currently being looked at, is the lightweight cipher. A lightweight cipher is a cipher designed for use on devices that have limitations, such as low processing power or latency, and can not handle the current encryption methods that were designed for devices like a desktop or server [8]. NIST is currently running a lightweight cryptography competition to choose the standard lightweight cipher going forward.

Within the lightweight ciphers, there are two designs we will analyse.
The first one is the cipher SATURNIN, which was designed using the wide trail strategy. The wide trail strategy attempts to design the round functions to limit the number of trails with a low weight. This is done by focusing a lot of resources on the linear step of the round functions to improve the diffusion of the trails [6]. If a cipher uses this strategy, the bounds for the differential probability of their trails become easy to verify. This cipher made it to the second round of the NIST lightweight ciphers competition.
The second design is used by the construction SPONGENT, which was designed using the principles of PRESENT and thus uses a PRESENT-like permutation. This permutation was used in the finalist ELEPHANT [1][10]. It only uses S-box layers and shuffle mechanisms. These two permutations are what we will analyse.

## 1.1 Related Work

Within *Thinking Outside the Superbox* Bordes et al. have analyzed alignment and clustering properties of SPONGENT and SATURNIN, as well as RIJNDAEL and XOODOO [4]. They have shown the importance of

alignment for the differential and linear propagation properties of a cipher or permutation. They also put forward that while aligned ciphers result in easy-to-verify bounds compared to unaligned ciphers, they do result in more clustering. We will be expanding on that paper and analyzing the expected differential probability (EDP) of the differentials, as well as the differential probability (DP) of the differential trails. Differentials do not take into account what path was taken within the permutation. Due to the differential trails including the path taken in the permutation, multiple trails can cluster in a differential. Both differentials and trails also have a limit on how spread out they can be, which causes them to clip and not spread out as well as would be expected. We will distinguish between when differential trails are *clustering* in differentials and when the weight is being *clipped* due to there being an upper bound on the possible weights.

## 1.2 Contributions

When examining a cipher or a permutation, differences in the input propagate into differences in the output. Inadequate diffusion within the cipher or permutation can cause these differences to be unevenly distributed. Both clustering and clipping affect how these differences are distributed. Because of this, we will be making a distinction between the property of clustering and that of clipping.
This brings us to the research question we try to answer:

*How do clustering and clipping affect SATURNIN and SPONGENT with respect to differential cryptanalysis?*

## 1.3 Outline

Within this paper, we will first show the notations and conventions. In Chapter 2, we will give a background into cryptanalysis and, more specifically, differential cryptanalysis. Chapter 3 will give an insight into the design of SPONGENT and SATURNIN. In Chapter 4, we will explain what weights, clustering, and clipping are and how they affect SPONGENT and SATURNIN. Then in Chapter 5, the EDPs of differentials and the DPs of differential trails will be calculated so that we can make the distinction clear between clustering and clipping. This will be compared to preceding work done in previous papers, mainly in *Thinking Outside the Superbox* [4].

## 1.4 Notation and Conventions

Because this thesis is a continuation on the work done by Bordes et al. [4], we will be following their already defined conventions which we will repeat

below.
Whenever we use indices, they always begin at zero. We write $k$ as a non-negative integer $k \in \mathbb{Z}_{\geqslant 0}$. We use k as a placeholder for any non-negative value. We also define $[0, k-1] = \{i \in \mathbb{Z}_{\geqslant 0} : 0 \leqslant i \leqslant k-1\}$.
For a given set $S$, #S represents its cardinality. A given equivalence relation on set S is represented by $\sim$. If $a \in S$, then the equivalence class can be represented by $[a]_\sim$. We write $e_i^k$, for the $i$th standard basis vector in $\mathbb{F}_2^k$. We write + for vector addition in $\mathbb{F}_2^k$.

A state $a$ is represented by a vector of $b$ bits. It is either a vector that the permutation is applied to or a difference. For such a state $a \in \mathbb{F}_2^b$, $a_i$ refers to its $i$th component.
The main task of this paper is studying permutations of the form

$$f : \mathbb{F}_2^b \to \mathbb{F}_2^b \tag{1.1}$$

A permutation is a bijection with an equal domain and codomain.
A fixed key perspective means that the key used for a block cipher is fixed to a specific value. By using a fixed-key perspective, the block ciphers used are transformed into permutations. We fix all key bits to zero. These permutations consist of lightweight round functions of the form

$$f = R_r \circ \cdots \circ R_1 \circ R_0 \tag{1.2}$$

With r $\in \mathbb{Z}_{>0}$.
We write $f[r] = R_r \circ \cdots \circ R_1 \circ R_0$. We write f[0]= id with id the identity function. Each round function $R$ is composed of three possible step functions:

- a linear transformation L
- a nonlinear transformation N
- a constant addition i.

A linear transformation L: $\mathbb{F}_2^b \to \mathbb{F}_2^b$ of $a$ can be written as L$(a) =$ M$a$ with M representing a matrix of the form $M \in \mathbb{F}_2^{b \times b}$. Such a linear transformation can be represented by a simple linear mapping between each state. This is not the case for a non-linear transformation. For the nonlinear transformation N, we write $n$ for the number of S-boxes of N and $m$ for the size of the S-boxes. Suppose $\mathrm{B}_j = jm, \ldots, (j+1)m - 1$. Permutations of the index space are written as $\tau : [0, b-1] \to [0, b-1]$. By shuffle, we mean a linear transformation $\pi : \mathbb{F}_2^b \to \mathbb{F}_2^b$ given by $\pi(a) = P_\tau a$, where $P_\tau$ is a permutation matrix associated with some $\tau$ obtained by permuting the columns of a $(b \times b)$ identity matrix according to $\tau$.

The index sets $B_i \subset [0, b-1]$ forms an ordered partition. A projection onto bits of $a$ indexed by the ordered partition $\mathrm{B}_i$ is written as:

$$P_i(a) : \mathbb{F}_2^b \to \mathbb{F}_2^{\#B_i} \tag{1.3}$$

Let $\Pi$ be a partition of the index space consisting of $k$ boxes of size $l$. A shuffle layer is called a $\Pi$-shuffle if the associated permutation matrix can be partitioned into $k$ identity matrices of dimension $(l \times l)$. For a step function to be $\Pi$-aligned, it must respect the boundaries of the boxes.

# Chapter 2

# Preliminaries

## 2.1 Cryptanalysis

A cryptographic permutation is built up of a combination of round functions, which themselves consist of step functions. The linear propagation properties of a permutation are analyzed by linear cryptanalysis, and the differential propagation properties are analyzed through differential cryptanalysis.

## 2.2 Differential Cryptanalysis

Differential cryptanalysis looks at differences in the output of a block cipher, given fixed differences in the input. It looks at how these differences propagate within the cipher. Differential cryptanalysis consists of a chosen-plaintext attack to apply these differences in the input and see how they propagate to the output of the ciphers [2].

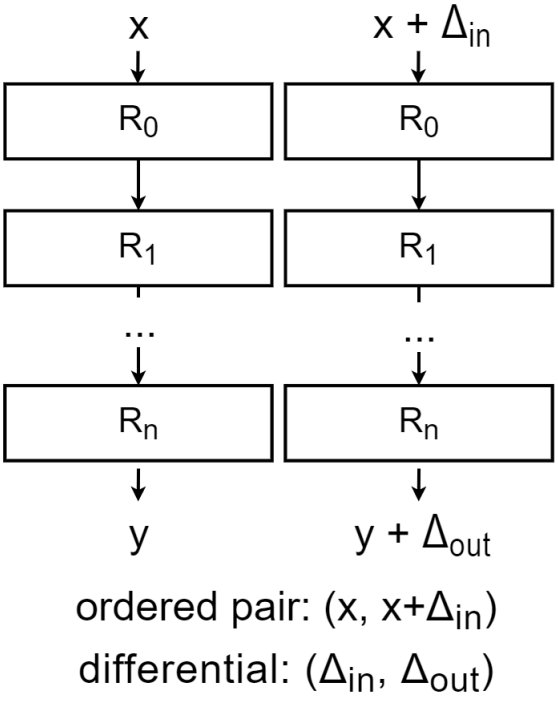


Figure 2.1: Differential propagation with input difference $\Delta_{in}$ and output difference $\Delta_{out}$.

To explain differential cryptanalysis, certain definitions first need to be explained. As this builds upon *Thinking Outside the Superbox*, we will be using many definitions and conventions introduced by them within section 2.1: Differential Cryptanalysis [4]. We will start off by explaining what a differential is and what the differential probability is. Then we will explain what the differential trails and the EDP are and how they can be calculated.

**Definition 1.** The *ordered pair* is the ordered pair of the input and the input with its difference $(x_{in}, x_{in} + \Delta_{in})$.

**Definition 2.** The *differential* is the ordered pair of the input difference and the output difference $(\Delta_{in}, \Delta_{out})$.

While the differential is also an ordered pair, going forward, we will be using *ordered pair* to refer to Definition 1, while *differential* will refer to Definition 2. Both are visualised in Figure 2.1.

Take an ordered pair $(x, x + \Delta_{in})$ with $f(x) + f(x + \Delta_{in}) = \Delta_{out}$. Then the ordered pair *follows* the differential $(\Delta_{in}, \Delta_{out})$. The ordered pairs that follow a differential form its solution set.

**Definition 3.** Let $f : \mathrm{F}_2^b \to \mathbb{F}_2^b$ be a permutation and define

$$V_f(\Delta_{in}, \Delta_{out}) = \{x \in \mathbb{F}_2^b : f(x) + f(x + \Delta_{in}) = \Delta_{out}\} \tag{2.1}$$

We call $V_f(\Delta_{in}, \Delta_{out})$ the *solution set* of the differential $(\Delta_{in}, \Delta_{out})$.

Any ordered pair $(x, x + \Delta_{in})$ with x $\in V_f$ $(\Delta_{in}, \Delta_{out})$ follows the differential $(\Delta_{in}, \Delta_{out})$. For $(\Delta_{in}, \Delta_{out})$ to be a *valid* differential, at least one ordered pair must follow it.

We now have the total amount of trails following a differential, but we would like to know the likelihood of a random ordered pair following a differential. This likelihood is the *differential probability* (DP) of the differential.
The DP of a differential is the number of ordered pairs in the solution set over the total number of possible ordered pairs.

**Definition 4.** The *differential probability* (DP) of a differential $(\Delta_{in}, \Delta_{out})$ over the permutation $f : \mathbb{F}_2^b \to \mathbb{F}_2^b$ is

$$\mathrm{DP}_f(\Delta_{in}, \Delta_{out}) = \frac{\#V_f(\Delta_{in}, \Delta_{out})}{2^b} \tag{2.2}$$

The differentials and the DP are the core of the research. The differential trails and the EDP build upon the differentials and the DP.

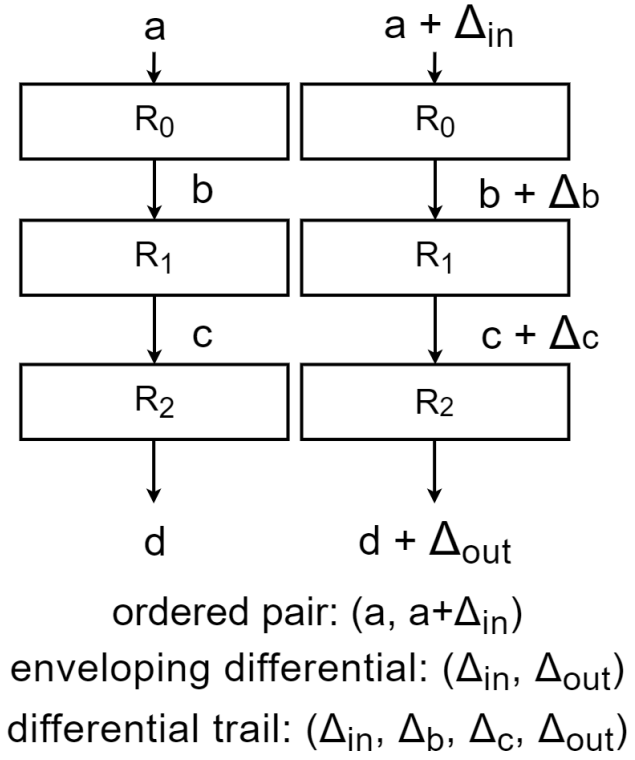


Figure 2.2: 3-round differential trail propagation with $a$ as input and $d$ as output.

The differential trail consists of the ordered pair of the input and output difference, just like the differential. However, it also consists of the sequence of intermediate differences in the permutation.

**Definition 5.** A sequence $Q = (q^{(0)}, q^{(1)}, ..., q^{(k)}) \in (\mathbb{F}_2^b)^{k+1}$ that satisfies $\mathrm{DP}_{R_i}(q^{(i)}, q^{(i+1)}) > 0$ for $0 \leqslant i \leqslant (k-1)$ is called a *k-round differential trail.*

For our purposes, the sequences we are investigating have the q values in between the round functions. This is displayed in Figure 2.3.

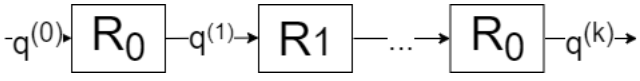


Figure 2.3: A trail Q.

Take an ordered pair $(x, x + \Delta_{in})$. Then the ordered pair is said to follow a differential trail $(q^{(0)}, .., q^{(k)})$ if all differences of the sequence it creates are the same as the differences in the differential trail. Any given ordered pair follows only one differential trail. This means that the trails partition the set of ordered pairs following a differential.

A differential trail follows the differential $(\Delta_{in}, \Delta_{out})$ if $q^{(0)} = \Delta_{in}$ and $q^{(k)} = \Delta_{out}$. We call $(\Delta_{in}, \Delta_{out})$ the *enveloping differential* of the trail. This is visualised in Figure 2.2.

The differential trail *core* consists of the values of the differential trail without the values of the differential $(\Delta_{in}, \Delta_{out})$ : $q^{(1)}, ..., q^{(k-1)}$.

$\mathrm{DT}(\Delta_{in}, \Delta_{out})$ represents the *set of differential trails in the (enveloping) differential* $(\Delta_{in}, \Delta_{out})$, with $\Delta_{in} = q^{(0)}$ and $\Delta_{out} = q^{(k)}$.
$\Delta_{out})$.

**Definition 6.** For any given differential $(\Delta_{in}, \Delta_{out})$, if

$$\#DT(\Delta_{in}, \Delta_{out}) > 1 \tag{2.3}$$

then there are multiple trails *clustering* in that differential.

An example is given in Figure 2.4.

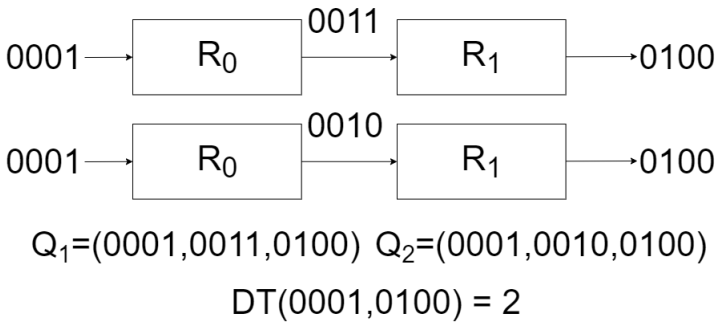


Figure 2.4: The trails $Q_1, Q_2$ clustering in differential (0001,0100).

Each round differential ($q^{(i)}$,$q^{(i+1)}$) has a solution set $V_{R_i}(q^{(i)}, q^{(i+1)})$. Take the transformed set of points $V_i = f[i]^{-1}(V_{R_i}(q^{(i)}, q^{(i+1)}))$ at the input of $f$. We say that an ordered pair $(x, x + q^{(0)})$ follow the differential trail if it holds that $x \in V_f(Q) = \bigcap_{i=0}^{k-1} V_i$. The fraction of states $a$ satisfying this equation is the DP of a trail.

**Definition 7.** The *differential probability* (DP) of a differential trail is

$$\mathrm{DP}_f(Q) = \frac{\#V_f(Q)}{2^b} \tag{2.4}$$

Given that the permutation consists of round functions, you can specify a differential with their own DP over each round function.

Next to the DP, there is also the EDP. This is the expected differential potential for a differential over all keys instead of over a fixed key [7]. The EDP of the differential *trail* is the expected differential potential over all keys as well.

**Definition 8.** Assume the keys have length $k$. The *expected differential probability* (EDP) of a differential is the average DP of that differential over all keys.

$$EDP(\Delta_{in}, \Delta_{out}) = \sum_{key=0}^{2^k} \frac{DP_{key}(\Delta_{in}, \Delta_{out})}{2^k}. \tag{2.5}$$

In the case that the round differential probabilities act independently, then the DP and EDP are the same.

Now that we have the basic definitions, we need a method to do calculations using the DP and the EDP. Depending on the structure of our permutation, different methods may facilitate the calculation of the DP and EDP over the full width of the round function. For this, we have other definitions for the (E)DP of a permutation.

**Definition 9.** The round differentials in a trail are said to be *independent* if

$$\mathrm{DP}_f(Q) = \prod_{i=0}^{k-1} \mathrm{DP}_{\mathrm{R}i}(q^{(i)}, q^{(i+1)}). \tag{2.6}$$

**Definition 10.** The *DP of the differential* is the sum of the DPs of all differential trails with initial difference $\Delta_{in}$ and final difference $\Delta_{out}$.

$$\mathrm{DP}_f(\Delta_{in}, \Delta_{out}) = \sum_{Q \in \mathrm{DT}(\Delta_{in}, \Delta_{out})} \mathrm{DP}_f(Q). \tag{2.7}$$

**Definition 11.** The *EDP of a differential* is the sum of the EDP values of all the trails in that differential [7].

$$EDP(\Delta_{in}, \Delta_{out}) = \sum_{Q \in DT(\Delta_{in}, \Delta_{out})} EDP(Q). \tag{2.8}$$

Take the function $R = i \circ \mathrm{N} \circ \mathrm{L}$ consisting of a non-linear S-box layer

N and a linear layer L with the addition of a round constant $i$. The differential trail of a differential $(\Delta_{in}, \Delta_{out})$ over R can be defined by specifying the intermediate differences as b,c, which results in $(\Delta_{in}, b, c, \Delta_{out})$. The value b can be represented as $L(\Delta_{in})$ because of the linearity of L. Because a difference does not change when a constant is added, we know that $c = \Delta_{out}$. This implies that the differential $(\Delta_{in}, b, c, \Delta_{out})$ has only one trail with a DP equal to the DP of the differential $(\Delta_{in}, L^{-1}(\Delta_{out}))$ over the S-box layer. It means that the DP of a round differential R is the product of DP values of the S-box differentials.

$$\mathrm{DP}_R(\Delta_{in}, \Delta_{out}) = \prod_{0 \leqslant j < b} \mathrm{DP}_{S_j}(P_j(\Delta_{in})), P_j(L(\Delta_{out}))). \tag{2.9}$$

**Definition 12.** The *restriction weight* of a differential $(\Delta_{in}, \Delta_{out})$ that satisfies $\mathrm{DP}_f(\Delta_{in}, \Delta_{out}) > 0$ is defined as

$$w_r(\Delta_{in}, \Delta_{out}) = -log_2(\mathrm{DP}_f(\Delta_{in}, \Delta_{out})) \tag{2.10}$$

This also means that given the weight, you can easily find the DP as $\mathrm{DP} = 2^{-w}$. For the differential trail, we add up the weights of the round differentials.

**Definition 13.** The *restriction weight of a differential trail* $Q = (q^{(0)}, q^{(1)}, ..., q^{(k)})$ is defined as

$$w_r(Q) = \sum_{i=0}^{k-1} w_r(q^{(i)}, q^{(i+1)}) \tag{2.11}$$

The restriction weight of a differential trail corresponds to its EDP. It is possible to use this weight to get the DP in specific situations. If the round differentials of the differential trail are independent, then this weight also represents the DP. Then the sum of the weights of the round differentials corresponds to the product of the round DPs.

## 2.3 Alignment

If a permutation is aligned, it allows us to reason about the bounds of the DP of differential (trails) of that permutation. Alignment allows us to define

a superbox that can be convolved to get the (E)DP for a two-round propagation. This way, we do not need to define a complete system. Because of this, this section will explain what alignment is. This way, we can show that our permutations are aligned and reason about the DP of trails more easily.

For alignment, we will examine the non-linear layer N consisting of $n$ S-boxes of size m as displayed in Figure 2.5. These S-boxes are laid out in parallel. We again refer to *Thinking Outside the Superbox* [4].

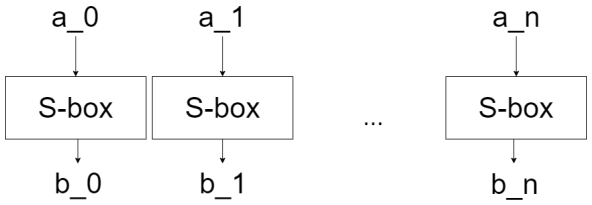


Figure 2.5: The non-linear layer N.

N consists of parallel applications of S-boxes to disjoint parts of the state indexed by $B_i$. We can write it as the following $\mathrm{S}_0 \times \mathrm{S}_1 \times .... \times \mathrm{S}_{n-1}$ with it characterized by

$$P_i \circ (\mathrm{S}_0 \times \mathrm{S}_1 \times ... \times \mathrm{S}_{n-1}) = \mathrm{S}_i \circ P_i \text{ for } 0 \leqslant i \leqslant n-1. \tag{2.12}$$

N then defines a unique ordered partition $\Pi_N = (B_0, ..., B_{n-1})$ of the index space $[0,\, b-1\,]$ with $\Pi_N$ being the box partition defined by N. We call the $B_i$ N-boxes. For a partition to be non-trivial, it must contain at least two elements.

Let $\Pi$ be a partition of the index space consisting of k boxes of size l.

**Definition 14.** We call $\Pi$ a *refinement* of $\Pi'$ and write $\Pi \leqslant \Pi'$ if for every $(i, B_i) \in \Pi$ there exists a $(j, B'_j) \in \Pi'$ such that $B_i \subseteq B'_j$.

**Definition 15.** We call $\phi : \mathbb{F}_2^b \to \mathbb{F}_2^b$ $\Pi$-*aligned* if we can decompose it as

$$\phi_0 \times \cdots \times \phi_{k-1} : \bigtimes_{i=0}^{k-1} \mathbb{F}_2^l \to \bigtimes_{i=0}^{k-1} \mathbb{F}_2^l \tag{2.13}$$

where $\phi_i$ represents the box functions.

**Definition 16.** Take a round function composed of parallel application N of equally sized S-boxes, a linear layer L, and an addition i of a constant. We say it is *aligned* if it is possible to decompose the linear layer L as L = M $\circ$ $\pi$ in such a way that:

- $\pi$ is a $\Pi_N$-shuffle
- M is aligned to a non-trivial partition $\Pi_M$ that satisfies $\Pi_N \leqslant \Pi_M$

If the round functions of a primitive are aligned, then the primitive is also aligned. Any primitive that is aligned has a superbox structure [9].

### 2.3.1 Superbox

Now take a two-round structure of the form: $\pi \circ M \circ N \circ \pi \circ M \circ N$ as seen in Figure 2.6. Since $\pi \circ M$ are linear steps, they will have no effect on the distribution. If we then define N' as $\pi^{-1} \circ N \circ \pi$, we can replace $N \circ \pi \circ M \circ N$ by $\pi \circ N' \circ M \circ N$ as $\Pi_N = \Pi_{N'}$. This then allows us to remove linear layer $\pi$. Because $\Pi'_N = \Pi_N \leqslant \Pi_M$, this can be viewed as a parallel application of superboxes. This is called a superbox layer. This results in our linear layer being $\pi \circ M \circ \pi$ and the non-linear layer being $N' \circ M \circ N$. For our two-round structure to be aligned, the non-linear layer has to be aligned to a non-trivial partition $\Pi$ such that $\Pi_N \leqslant \Pi_M$. Then the two-round structure is aligned to $\Pi_M$.

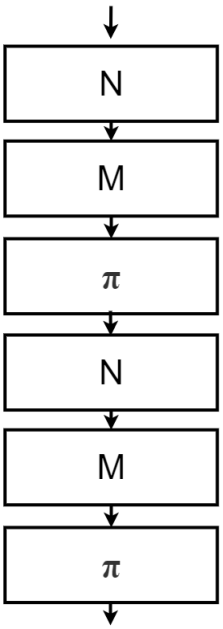


Figure 2.6: The two-round structure $\pi \circ M \circ N \circ \pi \circ M \circ N$.

# Chapter 3

# Saturnin and Spongent

This chapter will present the permutations that we will be investigating and explain how they are designed. While SATURNIN is a block cipher, it is turned into a permutation by taking a fixed key perspective and setting that key to zero.

## 3.1 Saturnin

The first permutation that we investigate is the one constructed by using SATURNIN [5].

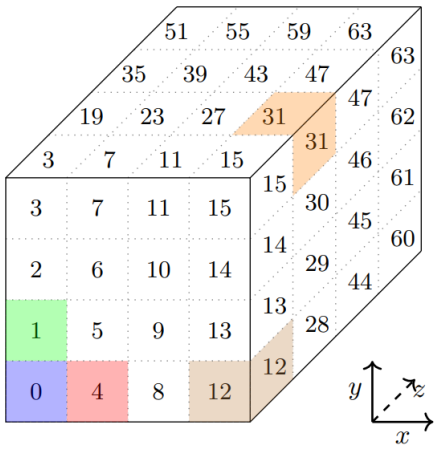


Figure 3.1: $4 \times 4 \times 4$ nibbles of SATURNIN [5].

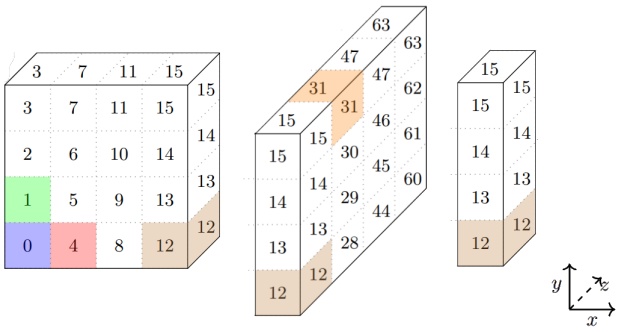


Figure 3.2: A SATURNIN slice, sheet and column respectively. Modified from [5].

It is a 256-bit block cipher with an equally sized key. It has three representations:

- A three-dimensional cube representation consisting of 4-bit nibbles in a $4 \times 4 \times 4$ cube represented in Figure 3.1.
- A two-dimensional representation consisting of 16-bit registers.
- A one-dimensional representation as 256 bits.

We will be using the three-dimensional cube representation going forward. The cube can be dissected into different sections, which are represented in Figure 3.2. A slice is a subset of the nibbles with z constant. A sheet is a subset of the nibbles with x constant. A column is a subset of the nibbles with x and z constant.

The actual algorithm that SATURNIN uses involves superrounds which consist of two (smaller) rounds. These smaller rounds consist of even and odd-numbered rounds. The even rounds only have one design, while the odd rounds have two designs. The designs are built up of 5 possible transformations.
A non-linear S-box layer S, which makes use of 2 different S-boxes forms, the non-linear transformations. A linear mixing layer MC and linear nibble permutation SR form the linear transformations. And a round constant addition RC and a round key addition RK form the constant additions.

The S-box layer applies a 4-bit S-box on all nibbles with an even index and a different 4-bit S-box on all nibbles with an odd index.
The MC layer uses 16 copies of a linear operation M over $(\mathbb{F}_2^4)^4$. This linear operation is applied to each column.
The nibble permutation SR is an operation that mixes the columns in each

of the sheets ($\text{SR}_{sheet}$) or the slices ($\text{SR}_{slice}$). For ($\text{SR}_{slice}$), the nibble coordinates are mapped from $(x, y, z)$ to $(x + y \mod 4, y, z)$. An example can be seen in Figure 3.3. For ($\text{SR}_{slice}$), the nibbles are mapped to $(x, y + z \mod 4, z)$.

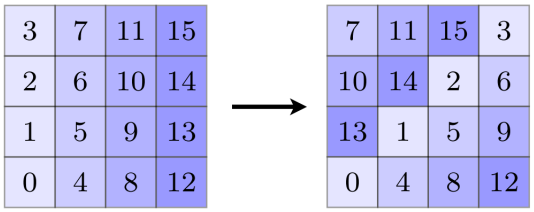


Figure 3.3: $\text{SR}_{slice}$ operation [5].

All even rounds consist of a linear and a non-linear layer: MC $\circ$ S.
The odd rounds use all transformations in the following structure: RC $\circ$ RK $\circ$ $\text{SR}^{-1}$ $\circ$ MC $\circ$ SR $\circ$ S.
Rounds 4r+1 uses an SR permutation using slices : RC $\circ$ RK $\circ$ $\text{SR}^{-1}_{slice}$ $\circ$ MC $\circ$ $\text{SR}_{slice}$ $\circ$ S.
Rounds 4r+3 uses an SR permutation using sheets RC $\circ$ RK $\circ$ $\text{SR}^{-1}_{sheet}$ $\circ$ MC $\circ$ $\text{SR}_{sheet}$ $\circ$ S.

We define the partition $\Pi_S$ so as to divide the state into 64 nibbles. Both nibble permutations created by the SR permutation are $\Pi_S$-shuffles. The mixing layer is aligned to a non-trivial partition $\Pi_{MC}$ that divides the state into 16 columns, each consisting of 4 nibbles. This then means that $\Pi_S \leqslant \Pi_{MC}$. Following Definition 16, SATURNIN is aligned.

For SATURNIN, there is not only a superbox but also a hyperbox. These hyperboxes consist of 4 superboxes put together and result in a width of 64 bits. These consist of the super-rounds that are a unique trait of SATURNIN. These super-rounds can be written in the form of the superbox layer consisting of S $\circ$ MC $\circ$ S and the linear layer $\text{SR}^{-1}_{slice}$ $\circ$ MC $\circ$ $\text{SR}^{-1}_{slice}$. The superbox layer is aligned to the partition $\Pi_{MC}$, while the linear layer is aligned to the non-trivial partition $\Pi_{slice}$.$\Pi_{slice}$ divides the state into four slices with four columns each, so $\Pi_{MC} \leqslant \Pi_{slice}$. The same holds for $\Pi_{sheet}$, which divides them into four sheets with $\Pi_{MC} \leqslant \Pi_{sheets}$. This means that the super-rounds are aligned and have their own superboxes, which are the hyperboxes.

The alignment properties are displayed in Figure 3.4.

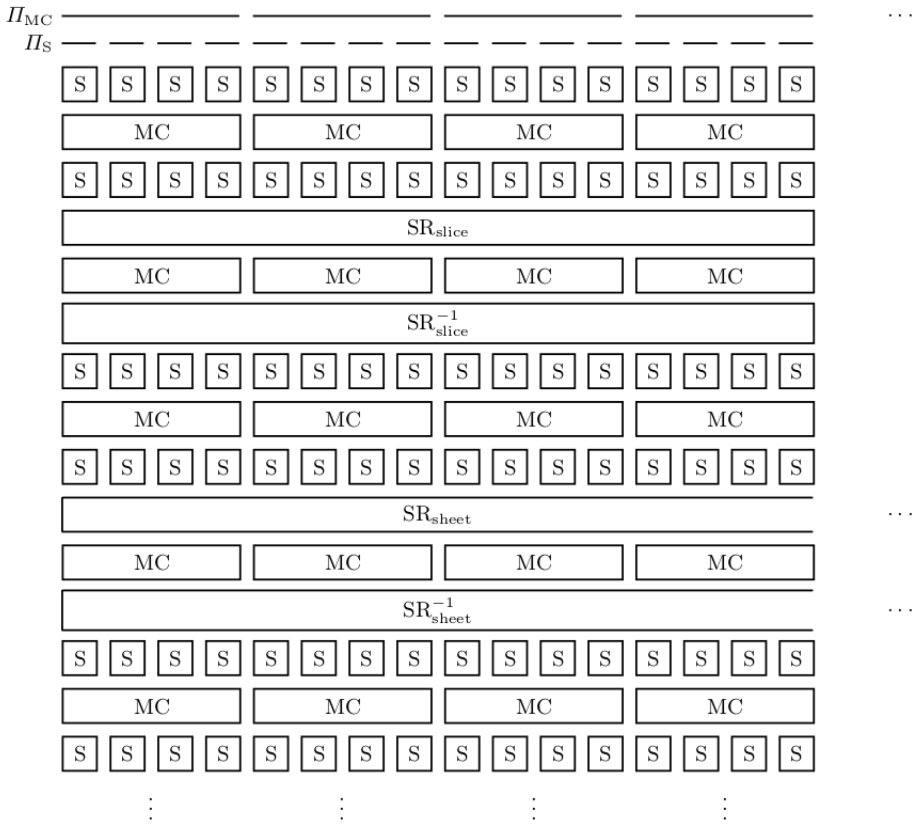


Figure 3.4: Alignment properties of SATURNIN [4].

## 3.2 Spongent

SPONGENT is a construction designed after the principles of PRESENT [3]. It uses a sponge construction that takes in a variable-length input and produces a variable-length output. This is displayed in Figure 3.5. Within SPONGENT, there is a permutation function $\pi_b$ that takes in a state of b bits. We will consider the case $b = 384$. These b bits consist of an inner layer of r bits and an outer layer of c bits. The bits in the inner layer of the state are XORed with the message, unlike the bits in the outer layer. These do not get affected until they join the first application of permutation $\pi_b$.

We are not interested in SPONGENT itself however, but in the permutation $\pi_b$ within SPONGENT.

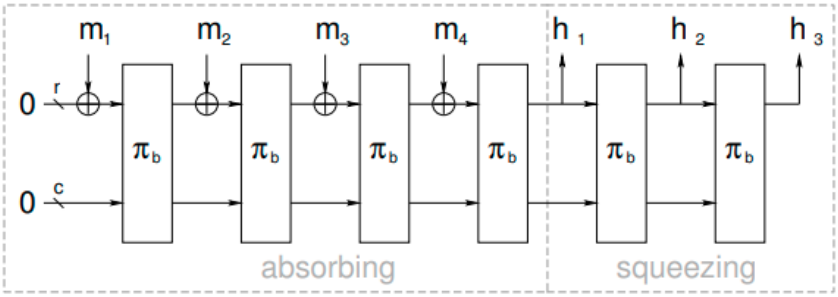


Figure 3.5: Construction of SPONGENT [3].

The permutation $\pi_b$ uses three different step functions to build its round function:

- a non-linear S-box layer *sBoxLayer*, consisting of 4-bit S-boxes with $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$
- a round constant addition *lCounter* that uses a $[log_2 R]$-bit LFSR
- a bit shuffle *pLayer*

$$pLayer(j) = \begin{cases} 96 \text{ j mod } 383 & \text{if } j \in [0, 382] \\ 383 & \text{if } j = 383 \end{cases}$$

These are used in the following structure:

for $i = 1$ to R do
$state \leftarrow lCounter_b^{reverse}(i) \bigoplus state \bigoplus lCounter_b(i)$
$state \leftarrow sBoxLayer_b(state)$
$state \leftarrow pLayer_b(state)$

The pLayer can be deconstructed into a linear mixing layer *SpongentMixLayer* followed by a box shuffle *SpongentBoxShuffle*.
*SpongentMixLayer* applies the mixing function *SpongentMix* to 24 subgroups of S-boxes [3]. Each subgroup consists of 4 S-boxes and is 16 bits wide. Each group consists of 4 subgroups and is 64-bits wide. The grouping is displayed in Figure 3.6. This mixing function is a bit shuffle that is associated with an index permutation on a subgroup. *SpongentBoxShuffle* is a box shuffle associated with a box index permutation of a box with a width of 96.

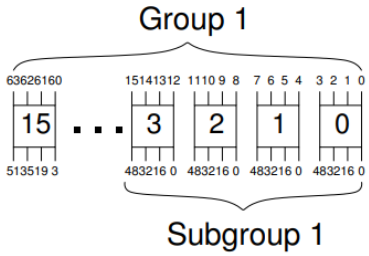


Figure 3.6: Groups and subgroups of S-boxes operated on by pLayer [3].

The *sBoxLayer* defines the box-partition $\Pi_{sBoxLayer}$ corresponding to 96 4-bit boxes. The *SpongentMixLayer* is aligned to the non-trivial partition $\Pi_{SpongentMixLayer}$ that splits into 96 16-bit subgroups [4]. So $\Pi_{sBoxLayer} \leqslant \Pi_{SpongentMixLayer}$. This means that SPONGENT $\pi_b$ itself is aligned, and we can define a superbox that can be convolved into the complete permutation. The alignment properties of SPONGENT $\pi_b$ are represented in Figure 3.7.

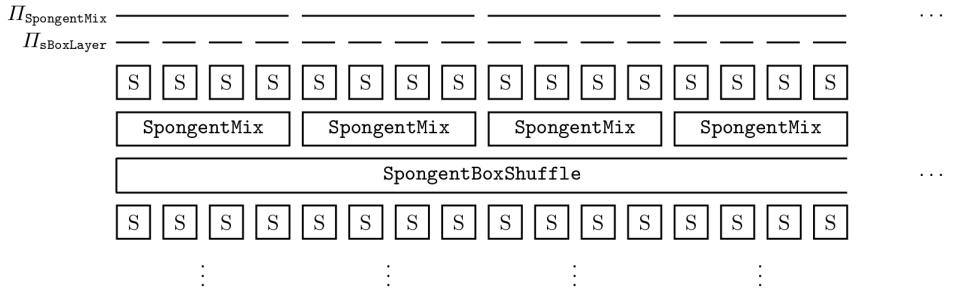


Figure 3.7: Alignment properties of SPONGENT $\pi_b$ [4].

## 3.3 Superbox Setup

The superboxes of SATURNIN and SPONGENT have already been defined in Sections 3.2 and 3.1. They are of the form $f = N \circ L \circ N$.

The 16-bit values are represented by an uppercase letter e.g. $A$ that represents the complete state, $A = a_0, a_1, a_2, a_3$ with $a_i$ representing the nibbles going into the $i$-th S-box. This is because the design of the superbox consists of four 4-bit S-boxes for the non-linear layer.

The first and second non-linear layers are both S-box layers (S). The SATURNIN superbox uses two distinct types of S-boxes. Each S-box layer consists of two S-box functions: S-box_even and S-box_odd. When a 16-bit input is entered as $x_0, x_1, x_2, x_3$, the bits in $x_0$ and $x_2$ go through S-box_even and $x_1$ and $x_3$ go through S-box_odd. This is visualised in Figure 3.8.

Both SATURNIN and SPONGENT use their own linear layers.
The SATURNIN superbox uses a mixing layer (M) : $SA = S \circ M \circ S$ while SPONGENT uses a shuffle layer (SH) : $SP = S \circ SH \circ S$.
The basic setups of the superboxes are displayed in Figure 3.9.

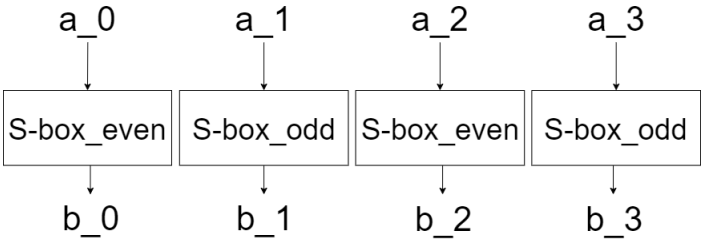


Figure 3.8: SATURNIN S-box layer.

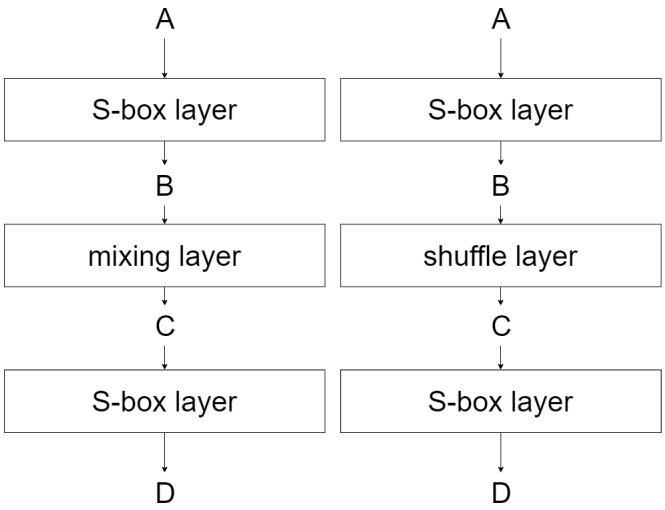


Figure 3.9: Setup of the SATURNIN superbox on the left and the SPONGENT superbox on the right.

# Chapter 4

# Research Basis

The data that has been gathered will be displayed as clustering histograms. These histograms show us how much the active bits or boxes cluster together. We will explain what bit and box weights are and what it means to measure their weight.

## 4.1 Bit and Box Weights

To understand bit and box weights, we will again be referring to definitions from *Thinking Outside the Superbox* [4].

When measuring the bit and box weights, we measure the weights for a bit or box activity. For this, we will first define an indicator function. We define the indicator function $1_i : \mathbb{F}_2^b \to \mathbb{F}_2$ with respect to the box partition $\Pi$ as

$$1_i(a) = \begin{cases} 0 \text{ if } P_i(a) = 0 \\ 1 \text{ otherwise} \end{cases}$$

Take a difference or linear mask $a \in \mathbb{F}_2^b$. A bit $i$ is *active* in $a$ if $a_i = 1$. The bit weight of $a$ is then given by the number of active bits. The bit weight is defined by

$$w_2(a) = \#\{i \in [0, b-1] : a_i \neq 0\}. \tag{4.1}$$

Take a difference or linear mask $a \in \mathbb{F}_2^b$. A box $B_i$ is *active* in $a$ if $1_i(a) = 1$ holds. Otherwise it is passive. The box weight is determined by the number of active boxes. If there is a non-zero bit in a box, then this box is considered active. The box weight of $a$ is defined by

$$w_{\prod}(a) = \#\{i \in [0, b-1] : 1_i(a) \neq 0\} \tag{4.2}$$

The activity pattern of a is defined by

$$r_\Pi(a) = \sum_{i=0}^{n-1} 1_{B_1}(a) e_i^n \tag{4.3}$$

It is the vector whose $i$th component is one if the box $B_i$ is active and zero otherwise.

Take a two-round trail $N \circ L \circ N$ with $(\Delta_{in}, b, c, \Delta_{out})$. The box weight of this two-round trail can be bounded from below by the sum of the number of active S-boxes at the input and the output of L. This is because in differential trails, c = L(b).

## 4.2 Clustering Histogram

We will now look at how this activity clusters together before and after a transformation. For this, we need to understand cluster-equivalence. Cluster-equivalence is merely a refinement of the box-activity-equivalence. This happens when two different states have the same activity relative to a box partition. Out of the box activity equivalence and Lemma 1, we get our definition of cluster-equivalence. Lemma 1 and the necessary definitions are displayed below [4].

**Definition 17.** Two states are *box-activity equivalent* if they have the same activity pattern with respect to a box partition Π:

$$a \sim a' \text{ if and only if } r_\Pi(a) = r_\Pi(a'). \tag{4.4}$$

The box activity class of $a$ is the set of states that are box-activity equivalent with $a$. They are represented as $[a]_\sim$.

**Lemma 1** Two trail cores ( $a_0$,$b_0$, ... ,$a_{r-2}$,$b_{r-2}$) and ( $a_0^*$,$b_0^*$, ... ,$a_{r-2}^*$,$b_{r-2}^*$) over a function $f = N_{r-1} \circ L_{r-2} \circ N_{r-2} \circ ... \circ L_0 \circ N_0$ that are in the same differential satisfy $a_0 \sim a_0^*$ and $b_{r-2} \sim b_{r-2}^*$.

Lemma 1 has already been proven before [4]. By taking the case of r = 2 in Lemma 1, we can refine the box-activity equivalence to the cluster-equivalence. Cluster-equivalence is shown in Figure 4.1.

**Definition 18.** Two states are *cluster-equivalent* with respect to a linear mapping L: $\mathbb{F}_2^b \to \mathbb{F}_2^b$ and a box partition $\Pi$ if they are box-activity equivalent before L and after it:

$$a \approx a' \text{ if and only if } a \sim a' \text{ and } L(a) \sim L(a'). \tag{4.5}$$

The cluster class of $a$ is the set of states that are cluster-equivalent with $a$. They are represented as $[a]_\approx$. The cluster partition is the partition of $\mathbb{F}_2^b$ to these cluster classes.

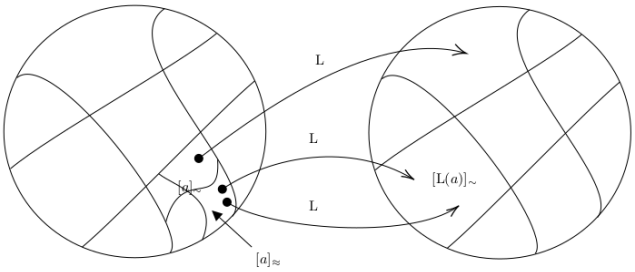


Figure 4.1: Partitions of $\mathbb{F}_2^b$ defined by $\sim$ and $\approx$ [4].

**Corollary 1** If two two-round trail cores (a,L(a)) and (a*, L(a*)) over f = $N \circ L \circ N$ are in the same differential, then $a \approx a^*$.

This corollary has already been proven [4]. From this, it follows that differences of two-round trail cores cluster together in the same cluster class. If the cluster classes are small, then there is little clustering. If they are large, then there is a lot of clustering.

Finally, to get to the clustering histogram, we need to define the cluster weight. The weight is given by

$$\tilde{w}([a])_\approx : \text{For all } a' \in [a]_\approx : \text{ box weight } w_\prod(a') + w_\prod(L(a')) \text{ is the same.} \tag{4.6}$$

**Definition 19.** Let L: $\mathbb{F}_2^b \to \mathbb{F}_2^b$ be a linear transformation and let $\approx$ be the relation given by Definition 18. The *cluster histogram* $\mathrm{N}_{\Pi,L} : \mathbb{Z}_{\geqslant 0} \times \mathbb{Z}_{\geqslant 0} \to \mathbb{Z}_{\geqslant 0}$ of $L$ with respect to the box partition $\Pi$ is given by

$$N_{\Pi,L}(k,c) = \#\{[a]_\approx \in \mathbb{F}_2^b/\approx : \tilde{w}([a])_\approx = k \wedge \#[a]_\approx = c\} \tag{4.7}$$

For a fixed box weight, the cluster histogram shows the distribution of the cluster classes with that box weight. Large cluster classes with small weight may result in two-round trails with a large DP. So for small weights, small cluster classes are better.

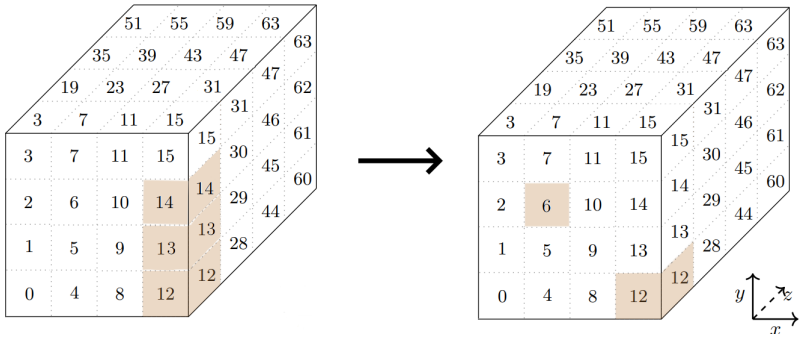


Figure 4.2: A simplified illustration of clustering with SATURNIN $\tilde{w} = 5$ after an $SR_{slice}$ transformation. Modified from [5].

The resulting cluster histograms for SATURNIN and SPONGENT can be seen in Figures 4.3 and 4.4. Here, C represents the cardinality of the equivalence class with respect to relation $\approx$. $N$ Represents the number of equivalence classes with that cardinality. Figure 4.2 shows a simplified example of cluster equivalence for SATURNIN.

| $\widetilde{\mathrm{w}}$ | $N \times C_{m,n}$ SATURNIN superbox |
|---|---|
| 5 | $(56 \times 15)$ |
| 6 | $(28 \times 165)$ |
| 7 | $(8 \times 2625)$ |
| 8 | $(1 \times 39075)$ |

Figure 4.3: Cluster histogram of SATURNIN superbox [4].

| $\widetilde{\mathrm{w}}$ | $N \times C$ |
|---|---|
| 2 | $(16 \times 1)$ |
| 3 | $(48 \times 1)$ |
| 4 | $(32 \times 1)\,(36 \times 7)$ |
| 5 | $(8 \times 1)\,(48 \times 25)$ |
| 6 | $(12 \times 79)\,(16 \times 265)$ |
| 7 | $(8 \times 2161)$ |
| 8 | $(1 \times 41503)$ |

Figure 4.4: Cluster histogram of Spongentmix SPONGENT [4].

## 4.3 Clustering and Clipping

### 4.3.1 Clustering

As we have shown before in Chapter 2.2 and in Figure 2.4, trails can cluster together in enveloping differentials. Clustering happens a lot less in differentials with less active S-boxes. The more active S-boxes there are, the more trails will cluster. Given our 16-bit superboxes, this would result in a lot of differences being found around weight 16.

### 4.3.2 Clipping

There is more going on than only clustering. Next to clustering, the DP also experiences clipping. The DP uses a fixed key, while the EDP does not. This means that the amount of ordered pairs the EDP can have is multiplied by the number of keys, while the DP only uses one key. This is the first factor that results in clipping of the DP. The second one is that the differences are not signed. This means that for any given ordered pair $(x, x + \Delta_{in})$ that follows the differential or the differential trail, there will also be $(x + \Delta_{in}, x)$ following it. This results in every differential over a superbox having an even number of pairs. This also means that for a $k$-bit superbox, there are only $k - 1$ different pairs, which limits the DP.

**Lemma 2** A trail over a $b$-bit superbox has a DP that is a multiple of $2^{1-b}$.

For our chosen superboxes, this means the following: Because the superboxes of SATURNIN and SPONGENT are 16 bits wide, the highest weight the DP can have is also 16, before considering that the differences are unsigned. Because the differences are not signed, the highest weight is actually 15. The weights for the DP of the differentials and the EDP of the differential trails have already been calculated in the past [4]. They can be seen in Figure 4.5.

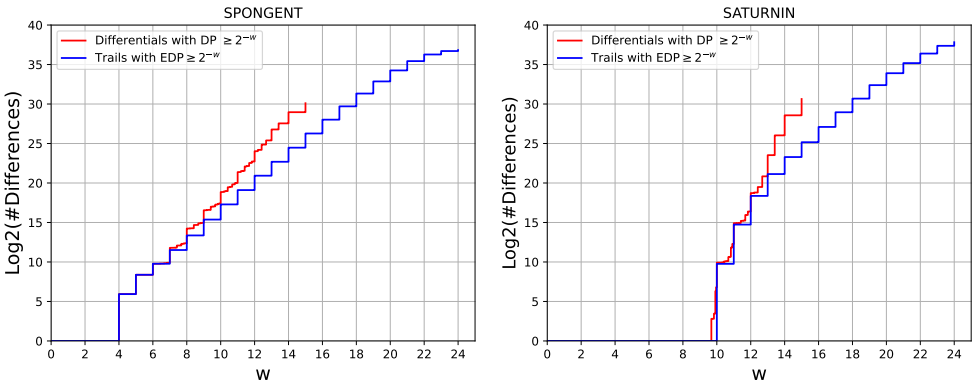


Figure 4.5: Differentials DP and differential trails EDP in the superboxes of SPONGENT and SATURNIN [4].

Within Figure 4.5, we can see the cumulative weight histograms of the differentials and the differential trails for both the SATURNIN and the SPONGENT superboxes. Within the cumulative histogram, for any given weight $w$, we see the number of differences with weight w or less. So any consecutive weight $w_i$ consists of the previous weight $w_{i-1}$ with the differences $y_i$ with weight $w_i$ added on top.

$$w_i = w_{i-1} + y_i \tag{4.8}$$

Both start closely together, and as the weight grows, they start diverging. This is the result of clustering as well as clipping.
We can see that SATURNIN has more divergence than SPONGENT. SATURNIN has 50x more differentials DPs than trails EDPs at weight 15 or less compared to SPONGENTs 20x more.

# Chapter 5

# Research Results

We will now build on the basis given in Chapter 4 by measuring how much the two-round trails of each permutation are experiencing clustering and how much they are experiencing clipping. We then compare the information we obtain with previous works on the DP of differentials and the EDP of differential trails.

## 5.1 DP of Differential Trails

There is already a method to calculate the DP of differentials $(\Delta A, \Delta D)$ [4]. That method traced in which differentials the input differences for all pairs ended up at. We will be expanding on that by adding a distinction for the intermediary differences of differentials, so that we are then tracing what differential trails they end up at. For our chosen superboxes, this means that we have to consider the intermediary differences $(\Delta B, \Delta C)$ as well. But between these two values, there is a linear layer. This means that the value of one infers the value of the other one. So we only need one of the two values to calculate the trails DP. We have chosen to calculate the value $\Delta B$ for a trail consisting of the values $(\Delta A, \Delta B, \Delta D)$.

```
forall A{
        forall X{
                delta_B= S-boxlayer(A) + S-boxlayer(A+X)
                delta_D= superbox(A) + superbox(A+X)
                trails[X]= (delta_B,delta_D)
        }
        sort(trails)
        count = 0
        forall X{
                if (trails[X]== trails[X-1]) count ++
                else histogram[count]++,
                    count=1
        }
}
```

Figure 5.1: Algorithm for computing the DP of differential trails.

The algorithm used to compute the DP of the differential trails is shown in Figure 5.1. To do the calculation, all possible values for $A$, $B$ and $D$ could be iterated over. While this is the most straightforward method, it is slow and inefficient. This can be improved by iterating over the ordered pair $(A, A + X)$ instead. We would now only need to calculate the values for $A$ and $X$. This results in only needing $2^{32}$ iterations instead of $2^{48}$.

As we are interested in the *difference* and not the results themselves, we can use the $XOR$ function to only keep the bits that are different after running input $A$ and $A + X$ through the necessary layers. To get the difference in $B$, we $XOR$ the results of running the ordered pair through the S-box layer. To get the difference in $D$, we $XOR$ the results of running the ordered pair through the entire superbox.

Once we have found the $\Delta B$ and $\Delta D$ values, we need to add them as a valid trail of $(\Delta A, \Delta B, \Delta D)$. We opted for a 1-dimensional array $M = bd$ with bd consisting of a bit shifted b added to the d, so they are stored within a single value. However, we do not need the trails themselves, but how often they appear. To count how often each trail appears, we sort the array of trails. This results in trails with the same values appearing next to each other. We can then iterate through the array to count how often a trail appears in a row. Any time the value in the array changes, we will know how many times a certain trail $(\Delta A, \Delta B, \Delta D)$ showed up, and we reset our counter for the new trail. The histogram array counting how often a certain count was found does not need to be reset, as it will simply get overwritten on the next iteration over A.

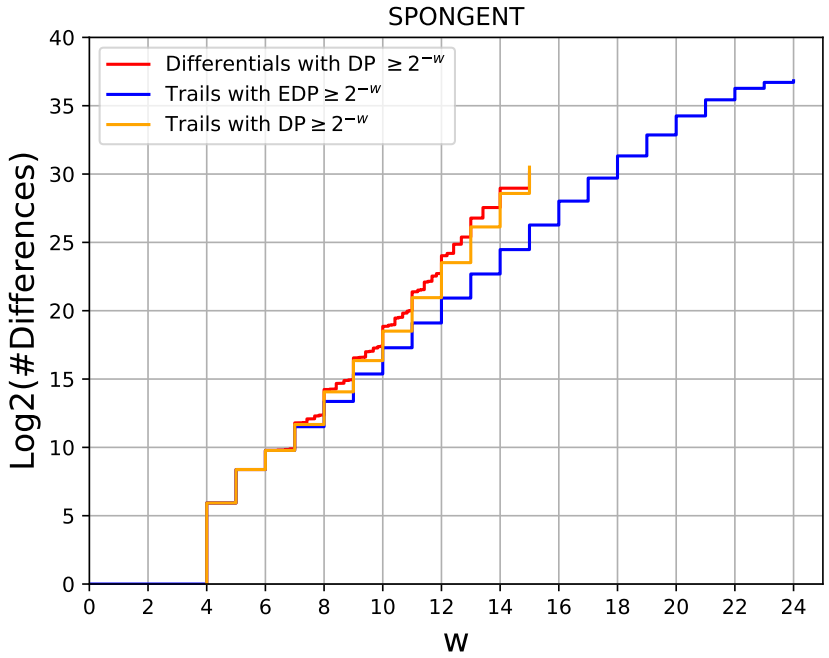


Figure 5.2: Trails DP with differentials DP and trails EDP of SPONGENT.

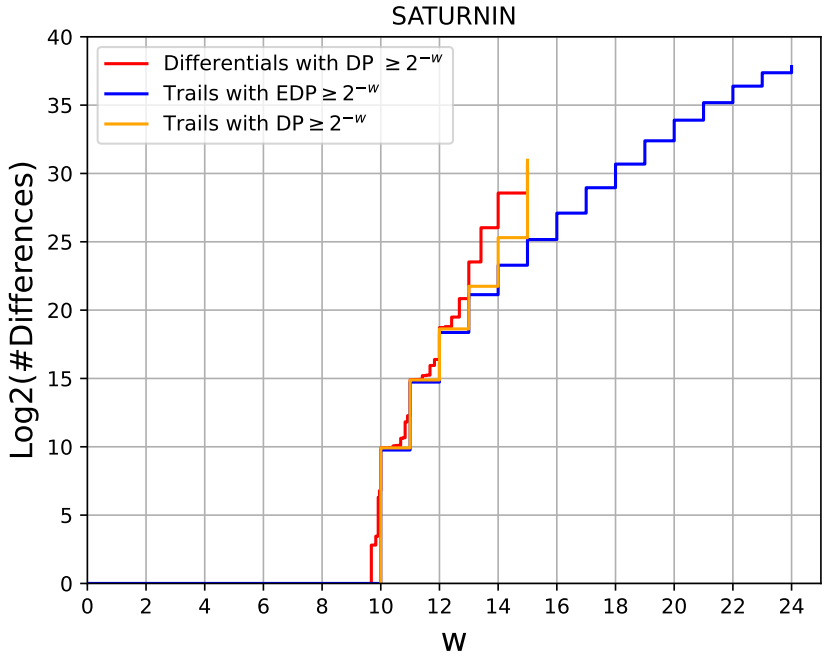


Figure 5.3: Trails DP with differentials DP and trails EDP of SATURNIN.

Figures 5.2 and 5.3 represent the differential trails DP(trails DP). We see that the curves end at weight 15 as is expected of clipping.
For SPONGENT, we can see that the trails DP starts at the same weight and fits within the data for the differential trails EDP and differentials DP that had already been calculated. The trails DP also grows consistently between the differentials DP and the trails EDP. We see that the trails DP curve is consistently a little below the differentials DP, and they end up merging at the same point at weight 15.

For SATURNIN, the weight of our trails DP starts at restriction weight 10, while the differentials DP starts slightly earlier. We also see a more significant difference in the number of pairs before they merge at weight 15 compared to SPONGENT. This can be seen at weight 14 or less. For SPONGENT, there are about two times more differentials DP compared to the trails DP. This difference is roughly 30 times as large for SATURNIN.

This shows us that for the differential trails, the DP does not fall below the EDP and is constantly lower than what would be expected because of clipping. We can also see that SPONGENT gets impacted more throughout all the weights by clipping than SATURNIN.

## 5.2 EDP of Differentials

The EDP of the differentials can be calculated to make clustering clear within the permutation. Calculating it however, is not trivial.

The EDP of a differential consists of the average DP over that differential over all keys. To find the EDP, we can use Definition 8. So if the round differential probabilities are the same, we simply need to calculate the DP, and we get the EDP. We start by analyzing the layers of our superbox SB:

$$SB = S \times L \times S \tag{5.1}$$

L represents the linear layer, and S represents the S-box layer. Because L is a linear layer, we can rewrite C as L(B) and we can avoid having to calculate the DP of the linear layer. Since we are now only working with S-box layers, and we know that they work independently, we can apply Definition 9 which tells us that the DP of the differential is the product of the DP of the S-box layers.

$$EDP_{SB}(A, D) = DP_S(A, B) \times DP_S(C, D) \tag{5.2}$$

We now need to find the DP of the two S-box layers separately. As we know that the S-box layer is a round differential, we can apply equation (2.9) to find the DP of each S-box layer. We then get

$$DP_S(A, B) = DP_{Sbox}(a_0, b_0) \times DP_{Sbox}(a_1, b_1) \times DP_{Sbox}(a_2, b_2) \times DP_{Sbox}(a_3, b_3) \tag{5.3}$$

The same applies to the DP of both S-box layers.

The DP over an S-box is easy to compute iteratively and is given in Definition 4. As our S-boxes are 4-bits wide, we can iterate over all inputs and count how often a specific differential appears.

```
1  for a0,a1,a2,a3{
2      for b0,b1,b2,b3{
3          for c0,c1,c2,c3{
4              for d0,d1,d2,d3{
5                  DP(AB)=DP(a0,b0)*DP(a1,b1)*DP(a2,b2)*DP(a3,b3)
6                  DP(CD)=DP(c0,d0)*DP(c1,d1)*DP(a2,b2)*DP(a3,b3)
7                  DPtrail=DP(a0,b0)*DP(c0,d0)
8                  DP_AD[d] += DPtrail
9              }
10         }
11     }
12     count[DP_AD[d]]++;
13 }
```

Figure 5.4: Algorithm for calculating EDP of differentials.

The algorithm used to compute the EDP of differentials is displayed in Figure 5.4. Within the code, we calculate the DP of the differentials by going over all the possible values for A, B, C, and D. Once we find a trail, we multiply the DP of the S-boxes as well as the DP of each layer. This happens for the first and the second S-box layers in lines 5 and 6, respectively. In line 7, we multiply the DP of the layers to get the DP of the differential trail in $(\Delta A, \Delta B, \Delta D)$. These trails get added up in line 8 into the table, which gives us the DP of the differential DP(A,D). Since the round differential probabilities are the same, the EDP and the DP are also the same. We then count how often we hit a specific count for the DP of the differential.

In Figures 5.5 and 5.6, we can see how the differentials EDP of SPONGENT compares to that of SATURNIN. We can see how for low weights, there is a lot more clustering for SPONGENT compared to SATURNIN as the differentials EDP separates itself from the trails EDP. At weight 14 or less, SATURNIN has almost as many differentials as differential trails, while SPONGENT has roughly five times as many differentials. Once we reach weight 16 however, SATURNIN has caught up and surpassed that ratio of SPONGENT. This means that SATURNIN makes up for its lower clustering at low weights with a big spike around weight 16. This is visible in the convex growth between weights 14-16. We also see that the EDP starts to flatten out at weight 16 for both SPONGENT and SATURNIN. This happens because of the limited width of their S-box layers, both at 16 bits, which causes clustering. Since they are limited to 16 bits, this results in a lot of trails with a weight of around 16 for both permutations. One more thing of note is how SATURNIN is practically done flattening out at weight 17, while it takes until weight 18 for SPONGENT.

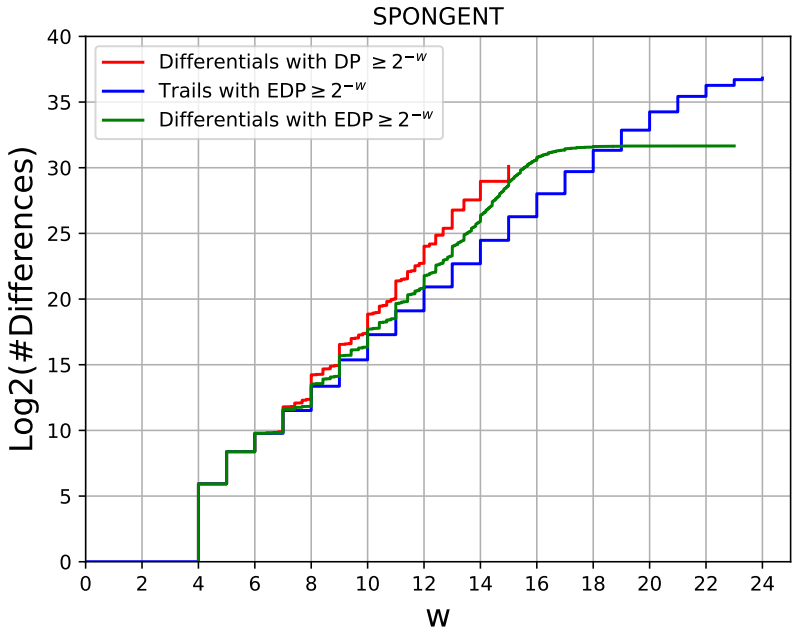


Figure 5.5: Trails DP with differentials DP and trails EDP of SPONGENT.

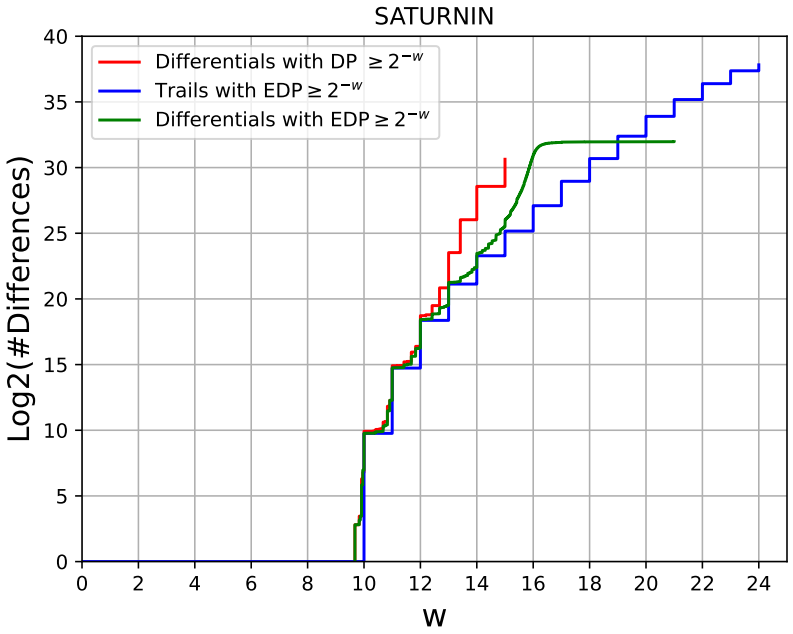


Figure 5.6: Trails DP with differentials DP and trails EDP of SATURNIN.

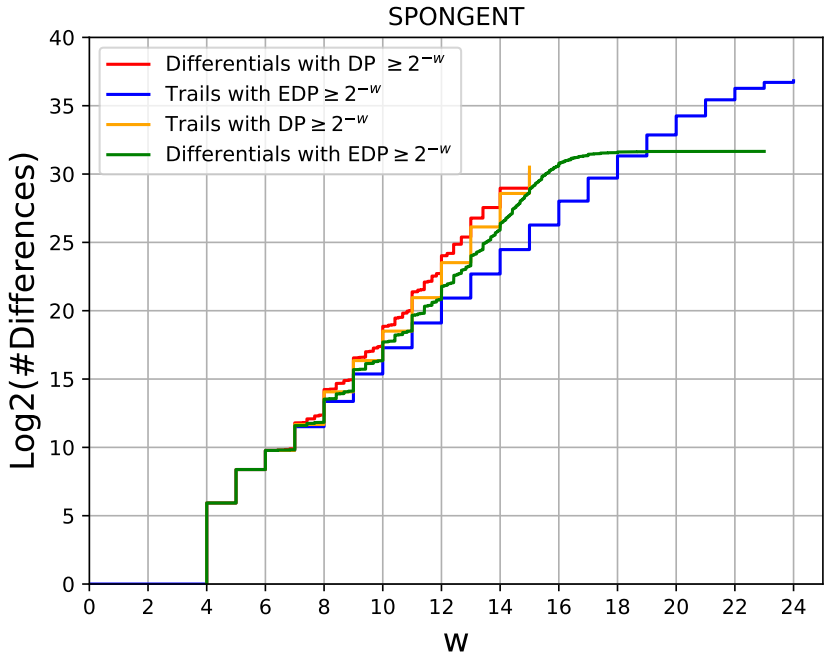


Figure 5.7: All DP and EDP curves of SPONGENT.

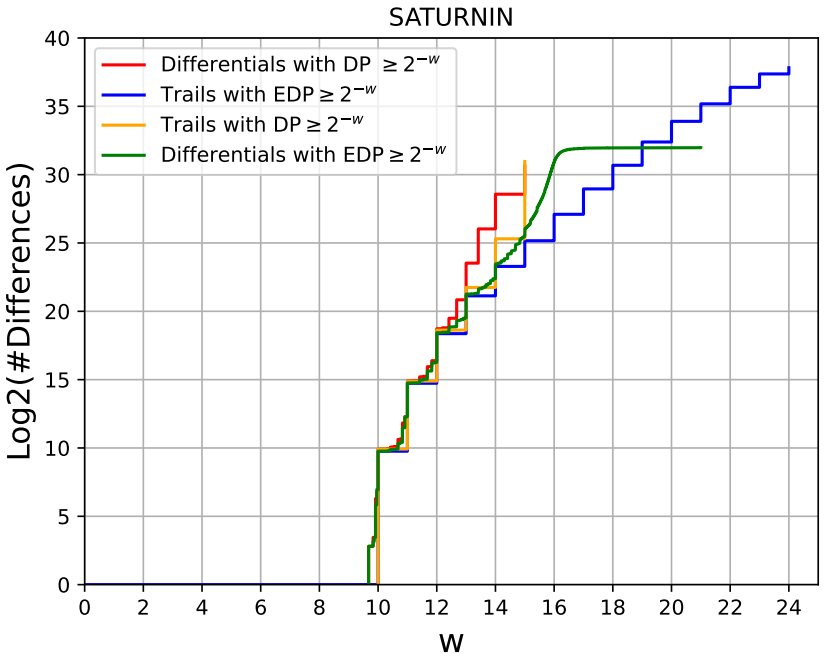


Figure 5.8: All DP and EDP curves SATURNIN.

In Figures 5.7 and 5.8, we can see the combined graph of all curves. Like with the original graphs from *Thinking Outside the Superbox* [4], we can again see the EDP curves staying above their respective DP curves. This is because the DP curves suffer from clipping while the EDP curves do not. The differentials are also always above their respective differential trails. They suffer from clustering while the trails do not.

The differentials EDP shows us the effects of clustering, and the trails DP shows clipping. Those together show us the differentials DP. It is important to note however that this is not because you can add a weight at the x-axis up to show clustering and clipping. Instead, clustering and clipping cause the high weights to fall to lower weights from the right side to the left side of the graph. This is best seen with SATURNIN.

## 5.3 Distribution of Pairs

In this section, we will compare the distribution of each of the curves by looking at what fraction of ordered pairs appear in each curve.
This will show us the following four curves:

1) The fraction of pairs in differentials with given DP.
2) The fraction of pairs in trails with given EDP, assuming independent difference propagation in the S-boxes.
3) The fraction of pairs in trails with given DP.
4) The fraction of pairs in differentials with given EDP, assuming independent difference propagation in the S-boxes.

To analyse the distribution, we will be putting everything in reference to the total number of ordered pairs. This is because every ordered pair only follows one differential or differential trail, so we can analyse the distribution rate up to the total number of ordered pairs.
For the DP values, this can be achieved by multiplying the number of differences per weight with the DP. For the EDP curves, we can multiply the number of differences with the EDP.
Then to get the percentage, we divide all differences by $2^{16}$. We then still need to take the log to make it more readable, which results in a graph with a negative y-axis going up to 0.
In Figures 5.9 and 5.10, we see the result of this in the cumulative graph.

We see how SATURNIN only starts getting affected by both clustering and clipping a lot at weight 13, while SPONGENT gets constantly affected by both clustering and clipping from around weight 9. We also see how for SATURNIN, clustering and clipping are mainly concentrated around weights

15 and 16, while for SPONGENT, they have a much more spread out effect. At weight 12 or less, 1/16th of the pairs have already been found for SPONGENT due to clustering and clipping. Without it, that would be only roughly 1/64th. Meanwhile, this is only around 1/512th for SATURNIN whether there is any clustering and clipping, or not. It is now also a lot more visible how most of the clustering for SATURNIN happens between weights 15 and 16.

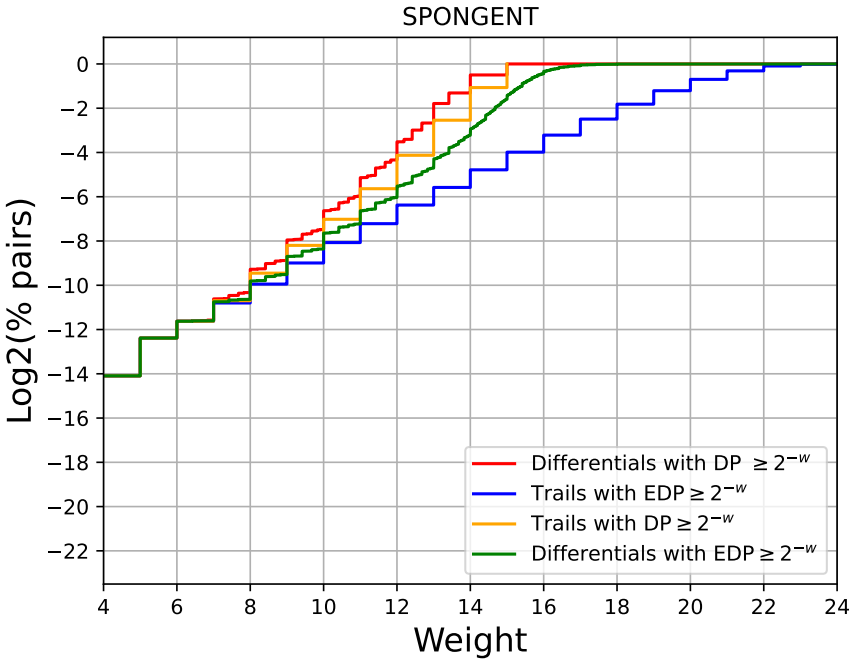


Figure 5.9: Distribution of pairs in SPONGENT.

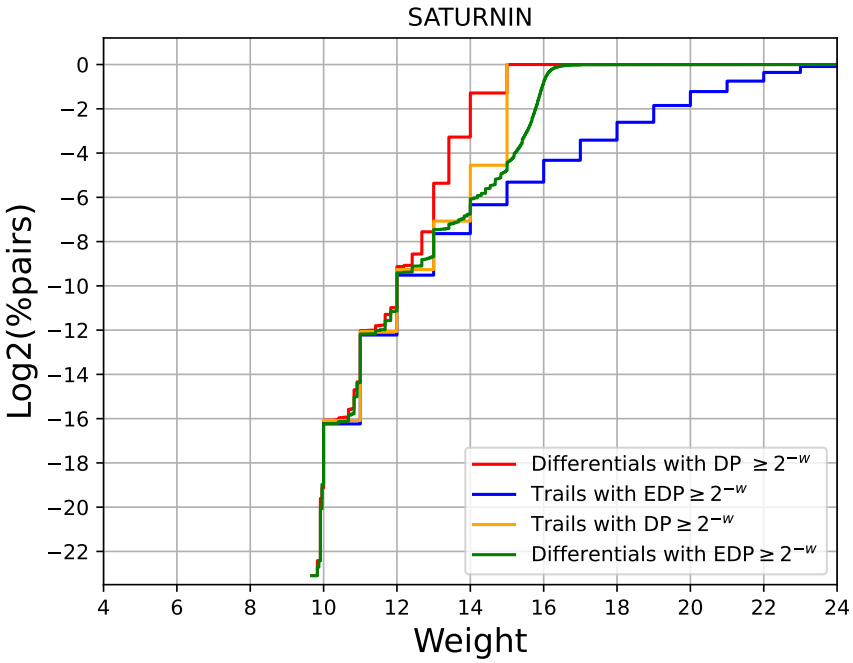


Figure 5.10: Distribution of pairs in SATURNIN.

# Chapter 6

# Conclusions

Before this thesis, we were only able to see the effects of clipping and clustering together. We have now separated these two effects. We have provided a method to calculate the EDP of differentials and the DP of differential trails for SATURNIN and SPONGENT. We have seen how SATURNIN suffers significantly from these two effects at higher weights, while SPONGENT experiences them more gradually through both lower and higher weights.

## 6.1 Future Work

We would suggest future work to analyse the effects on the linear trails of both SATURNIN and SPONGENT. It would also be interesting to analyse how different designs other than those used by SATURNIN or SPONGENT are affected by clustering and clipping.

# Bibliography


[1] Tim Beyne, Yu Long Chen, Christoph Dobrauning, and Bart Mennink. Elephant v2. 2021. `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/elephant-spec-final.pdf`.

[2] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology-CRYPTO' 90*, pages 2–21, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[3] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. Spongent: The design space of lightweight cryptographic hashing, 2011.

[4] Nicolas Bordes, Joan Daemen, Daniël Kuijsters, and Gilles Van Assche. Thinking outside the superbox. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 337–367, Cham, 2021. Springer International Publishing.

[5] Anne Canteaut, Sébastien Duval, Gaëtan Leurent, Maria Naya-Plasencia1, Léo Perrin, Thomas Pornin, and André Schrottenloher. Saturnin: a suite of lightweight symmetric algorithms for post-quantum security, 2019. `https://project.inria.fr/saturnin/files/2019/05/SATURNIN-spec.pdf`.

[6] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In Bahram Honary, editor, *Cryptography and Coding*, pages 222–238, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[7] Joan Daemen and Vincent Rijmen. Understanding two-round differentials in aes. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks*, pages 78–94, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[8] Kerry McKay, Lawrence Bassham, Meltem Sönmez Turan, and Nicky Mouha. Report on lightweight cryptography, 2017. `https://doi.org/10.6028/NIST.IR.8114`.

[9] Sangwoo Park, Soo Hak Sung, Seongtaek Chee, E-Joong Yoon, and Jongin Lim. On the security of rijndael-like structures against differential and linear cryptanalysis. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, pages 176–191, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[10] Meltem Sönmez Turan, Kerry McKay, Çağdaş Çalık, Donghoon Chang, and Lawrence Bassham. Status report on the first round of the nist lightweight cryptography standardization process, 2019. `https://doi.org/10.6028/NIST.IR.8268`.