IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2017, No. 3, pp. 1–23. DOI:10.13154/tosc.v2017.i3.1-23

Grøstl Distinguishing Attack: A New Rebound Attack of an AES-like Permutation

Victor Cauchois[1,2], Clément Gomez[1] and Reynald Lercier[1,2]

[1] Direction générale de l'armement - Maîtrise de l'information (DGA MI), Boîte Postale 7, 35998 Rennes Cedex 9, France
clement.gomez@m4x.org

[2] Institut de Recherche Mathématique de Rennes (IRMAR), Université de Rennes 1, Campus de Beaulieu, 35042 Rennes, France
{victor.cauchois,reynald.lercier}@m4x.org

Abstract.
We consider highly structured truncated differential paths to mount a new rebound attack on Grøstl-512, a hash functions based on two AES-like permutations, P_{1024} and Q_{1024}, with non-square input and output registers. We explain how such differential paths can be computed using a Mixed-Integer Linear Programming approach. Together with a SuperSBox description, this allows us to build a rebound attack with a 6-round inbound phase whereas classical rebound attacks have 4-round inbound phases. This yields the first distinguishing attack on a 11-round version of P_{1024} and Q_{1024} with about 2^{72} computations and a memory complexity of about 2^{56} bytes, to be compared with the 2^{96} computations required by the corresponding generic attack. Previous best results on this permutation reached 10 rounds with a computational complexity of about 2^{392} operations, to be compared with the 2^{448} computations required by the corresponding generic attack.

Keywords: Cryptanalysis · Hash function · Rebound attacks · AES-like · Grøstl

1 Introduction

Hash functions are of first importance in cryptography. Producing fixed size outputs from inputs of variable length, they are at the heart of many protocols to ensure integrity or authentication properties in numerous cryptographic applications, from signatures to encryption schemes. To be considered safe, they have to offer many security guarantees, among which protection against preimage, second preimage and collision attacks.

The indifferentiability framework, which became very popular among hash functions designers at the time of the SHA-3 competition, aims at avoiding these attacks. This security notion enables to instantiate cryptographic protocols proved secure in the random oracle model by a hash function. Hash functions that iterate a cryptographic permutation whose internal state is partially modified with the message during an absorbing phase and then partially extracted to build the output of the hash function during a squeezing phase are a large class of functions that can be proved secure in this model. However, their validities rely on the assumed ideal behavior of the permutation.

Distinguishers are algorithms that exhibit a non-randomness behavior of a permutation. Even when they do not directly weaken permutation-based hash functions, distinguishers break the security proof of the hash function. They are a first step in a cryptanalysis and reduce the confidence in these primitives.

Licensed under Creative Commons License CC-BY 4.0.
Received: 2017-06-01, Accepted: 2017-08-01, Published: 2017-09-19

Since Rijndael [DR02] has been chosen as the Advanced Encryption Standard, its “wide-trail strategy” design has inspired lots of symmetric-key primitives, especially since new processors have integrated the AES round function and key generation as basic processor operations. Rebound attacks, introduced in [MRST09], was the cryptanalists' response to the generalization of AES-like permutations used in hash function designs. It consists in finding a pair of internal register values in the middle of the permutation whose difference propagations towards the input and output registers follow a truncated differential path. When there exist efficient algorithms that yield such differences, it shows a non-random behavior of the permutation, and as such are distinguishers. A critical point is the choice of the truncated differential path, since the complexity of a rebound attack depends for a large part of it.

SHA-3 Cryptographic Hash Algorithm Competition organized by the National Institute of Standards and Technology (NIST) was the ideal playground to extend, develop and apply these techniques, as many of the candidates are built on top of AES-like permutations. Techniques as *start-from-the-middle*, introduced by [MPRS09], or *SuperSBox* descriptions, due to Daemen and Rijmen [DR06] and used for instance in [GP10, WFWS10, Gil14], are useful ideas to mount efficient rebound attacks. Further improvements due to [NP11] and [SLW$^+$10] have cleared the way for the most recent variants.

Grøstl [GKM$^+$09] is one of these AES inspired primitives. It has been one of the five finalists of the SHA-3 competition. Even if its mode of operation differs from the sponge construction [BDPVA08] used in Keccak, the function finally standardized by the the U.S. National Institute of Standards and Technology, its design still relies on two internal permutations and it comes with a security proof. Andreeva et al. [AMP10] prove that it is indifferentiable from a random oracle, up to the birthday bound under the assumed ideal behavior of the permutations. Two variants of Grøstl have been proposed by its authors, Grøstl-256 and Grøstl-512.

We focus on Grøstl-512, because its rectangular registers of 8×16 bytes make our attack possible on 11 rounds. In full generality, non square AES-like permutations are more vulnerable than square ones to rebound attacks, due to their slower diffusion, here three rounds instead of two rounds to obtain a full active state from a single difference, can be used to reach more rounds. The last rebound attack on Grøstl-512 in date is a work by Jean et al. [JNPP14]. It outlines that the rectangular shape of these permutations enables to extend the regular 9-round path of AES-like permutations to reach 10 rounds, thanks to a clever guess and determine algorithm. It turns out, that due to the dense differential pattern at the middle of the differential path, this 9-round path fully constrains the middle register value. In this paper, we consider instead a new differential path that spans over only half of the middle registers. In return, we obtain a huge number of register values that realize this differential path, and we take advantage of their easy characterization to filter among them these that yield a 11-round differential path.

This distinguisher applies on a reduced version of the two internal permutations, 11 rounds instead of the 14 rounds specified in [GKM$^+$09], and not on Grøstl-512 in its entirety. It is commonly admitted that such results help to evaluate the resistance of the Grøstl hash function, but as such, they do not threaten its security.

Although the presented truncated differential path is specific to Grøstl-512, we believe that the technique that we use to patch up three sets of differential values could be applied to other AES-like structures. Such a structured truncated differential path may be found as a solution to a Mixed-Integer Linear Programming problem (MILP). To the best of our knowledge, only few works make use of a MILP approach to analyze hash function, the first of which is a work by Bouillaguet et al. about the SHA-3 candidate SIMD [BFL11].

Acknowledgments. We wish to thank Henri Gilbert for comments that greatly improved the paper. We are also grateful to the referees for their constructive input.

Our Contributions

We present sparse and highly structured truncated differential paths for the permutations P_{1024} and Q_{1024} of Grøstl-512, obtained with Mixed-Integer Linear Programming techniques. We recast it as a SuperSBox description to mount a variant of the rebound attack with a 6-round inbound phase. This method allows us to obtain the best known complexities, reaching 11 rounds with a surprisingly low complexity of 2^{72} computations while previous results reached 10 rounds with a computational complexity of 2^{392}.

Table 1 summarizes the best known results on the 512-bit version of Grøstl. All these attacks are rebound ones, which mainly differ from the truncated differential path used.

Table 1: Known rebound attacks on Grøstl-512 internal permutations.

Rounds	Time	Generic attack	Memory	Reference
7	2^{152}	2^{512}	2^{56}	[SLW+10]
8	2^{280}	2^{448}	2^{64}	[JNPP14]
9	2^{328}	2^{384}	2^{64}	[JNPP14]
10	2^{392}	2^{448}	2^{64}	[JNPP14]
11	2^{72}	2^{96}	2^{56}	This paper (§ 4.2)

Outline of the Document

In Section 2, we present Grøstl-512 by focusing on its underlying permutation P_{1024}, detailing some properties of its building blocks. In Section 3, we present the structured truncated differential path that we found. We give a SuperSBox description of it and give some hints about its satisfiability and how it can be obtained. In Section 4, we present a distinguisher which finds a pair of inputs whose difference when propagated through the cryptographic permutation agrees with the truncated differential path. Every technical detail is then explained.

2 Description of Grøstl-512

Two different versions of Grøstl have been submitted to the SHA-3 hash function competition, Grøstl-256 which outputs 256-bit digests and Grøstl-512 which outputs 512-bit digests.

They handle messages by dividing them into blocks, using some padding. They update then iteratively an internal state initialized with some IV by computing a compression function f on both a block of message and the internal state as illustrated in Figure 1. Its design follows a wide-pipe construction: the size of the internal state is twice larger than the size of the output of the hash function. In both versions, the compression function is built on top of two AES-like permutations, they differ however in a fundamental way: Grøstl-256 uses a square matrix representation of its internal state whereas Grøstl-512 uses one with a rectangular matrix representation. We focus in this article on Grøstl-512.

2.1 Grøstl-512 Compression Function and Output Transformation

The compression function of Grøstl-512, f_{1024}, is built from two 1024-bit permutations P_{1024} and Q_{1024}, as illustrated in Figure 2a, according to the following definition,

$$f_{1024}(h, m) = P_{1024}(h \oplus m) \oplus Q_{1024}(m) \oplus h\,.$$

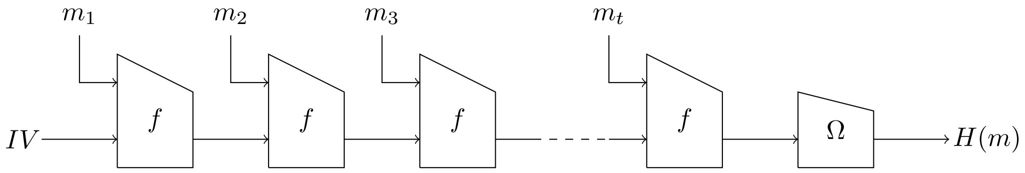
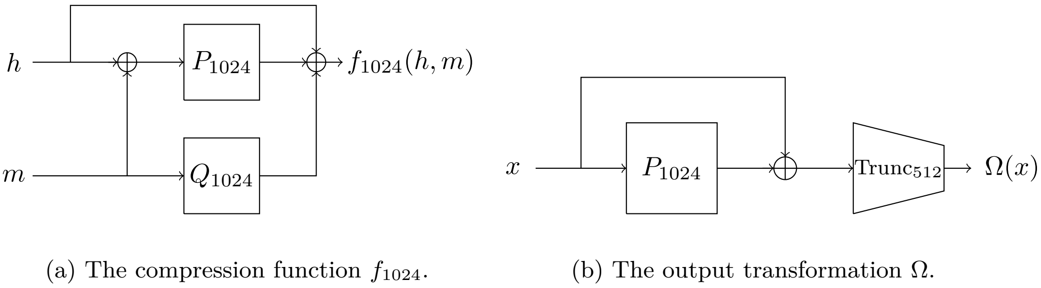


Figure 1: The Grøstl-512 hash function.

We denote by Trunc_{512} the truncation which returns only the 512 last bits of its input. The output transformation is simpler than the compression function and, as illustrated in Figure 2b, is defined by

$$\Omega(x) = \mathrm{Trunc}_{512}(P_{1024}(x) \oplus x)\,.$$



(a) The compression function f_{1024}.

(b) The output transformation Ω.

Figure 2: Grøstl-512 internal functions.

This compression function has been proved collision and preimage resistant under the assumption that P_{1024} and Q_{1024} are ideal [FSZ09]. Furthermore, the whole Grøstl-512 construction has been proved to be indifferentiable from a random oracle under this latter assumption and the additional hypotheses that P_{1024} and Q_{1024} are independent from each other [AMP10].

2.2 Grøstl-512 Internal Permutations Round Transformations

The 1024-bit internal states of the AES-like structures P_{1024} and Q_{1024} are specified as a 16×8 matrix of bytes. These permutations then consist in 14 iterations of the following round permutation,

$$R := \mathbf{MixBytes} \circ \mathbf{ShiftBytesWide} \circ \mathbf{SubBytes} \circ \mathbf{AddRoundConstant}\,,$$

where

- **AddRoundConstant** (**ARC**) adds a constant depending on the round to the internal state;
- **SubBytes** (**SB**) substitutes each byte in the matrix representation by its image by the non-linear **SBox** used in Rijndael. As it applies independently on bytes, we will refer as **SB** indifferently to consider this transformation on full states or on any partial states. To simplify our analysis, we consider an ideal behavior of this transformation:

$$\forall(\delta,\delta') \in (\mathbb{F}_{2^8}^*)^2,\ |\{X \in \mathbb{F}_{2^8} \,|\, \mathbf{SBox}(X) \oplus \mathbf{SBox}(X \oplus \delta) = \delta'\}| \in \{0,2\}\,. \quad (1)$$

Typical examples are almost perfect nonlinear (APN) functions [Dob99]. Another classical assumption due to the non-linearity of **SB** will be of great use:

$$\text{“The image of any set of distinct byte values by } \mathbf{SB} \text{ is uniformly distributed”} \qquad (2)$$

i.e. for all $Y_1 \neq Y_2 \neq \cdots \neq Y_k \in \mathbb{F}_{2^8}$,

$$\Pr_{X_1 \neq X_2 \neq \ldots \neq X_k}\{\mathbf{SBox}(X_1) = Y_1, \ldots, \mathbf{SBox}(X_k) = Y_k\} = \frac{k\,!}{2^8\,!}\,.$$

We assume that $\mathbf{SB}^{-1}$ behaves as its inverse;

- **ShiftBytesWide** (**Sh**) cyclically shifts the bytes within a row to the left by a number of positions in the matrix representation. For P_{1024}, Rows 1 to 8 are respectively shifted by 0, 1, 2, 3, 4, 5, 6 and 11 positions. For Q_{1024}, Rows 1 to 8 are respectively shifted by 1, 3, 5, 11, 0, 2, 4 and 6 positions;
- **MixBytes** (**MB**) applies to each column independently. We will refer as **MB** indifferently to consider this transformation on full states or on a single column. Bytes are seen as elements of $\mathbb{F}_{2^8}$. This transformation is built from a MDS matrix with coefficients in $\mathbb{F}_{2^8}$. This will have a great importance in our analysis, it means that the image of a column with $k > 0$ non-zero bytes by **MB** has non-zero bytes in at least $9 - k$ byte positions. The invert matrix of a MDS matrix being MDS, $\mathbf{MB}^{-1}$ behaves as its inverse.

Remark 1. The last round of AES-like structures traditionally avoids the **MB** transformation. To be consistent with rebound attacks literature, we will in this article always consider full rounds: with the **MB** transformation. A quick look at the truncated differential path that we use will however convince the reader that this change does not come into play in the attack.

3 A Distinguisher for Reduced-Round P_{1024} with 11 Rounds

We aim here to prove the non-randomness of a reduced version with 11 rounds of the permutations of Grøstl-512. To achieve this goal, we present a distinguisher, conflicting with a non-random behavior of these permutations. We focus here, arbitrarily, on the permutation P_{1024}, but everything can be made similarly for Q_{1024} (see Appendix A).

Remark 2. As it will be one fundamental measure of complexity in the remaining of this text, we denote $\beta = 2^8$. To to ease the presentation, we often use Landau asymptotic approximations $O()$, applied on powers of β to avoid additional logarithmic terms that typically arise from sorting operations. By a slight abuse of notation, we overload this notation for a fixed β, by assuring the reader that we have checked any “$O(\beta^n)$” to be smaller than β^{n+1}.

3.1 Limited-Birthday Distinguishers

Gilbert et al. introduced limited-birthday distinguishers [GP10]. The challenge consists in finding a pair of input values whose difference lies in a predefined subspace and whose images by the permutation have their difference lying in another predefined subspace.

Problem 1. ***Limited-birthday***(P, E_{in}, E_{out})*: Given a permutation P and two $\mathbb{F}_2$-linear subspaces E_{in} and E_{out}, find a pair of input values (X, X') such that $X \oplus X' \in E_{in}$ and $P(X) \oplus P(X') \in E_{out}$.*

They introduced the best known generic algorithm for solving this problem too, the so-called limited-birthday algorithm. Theorem 1 gives its complexity.

Theorem 1. *For a n-bit permutation P, a $\mathbb{F}_2$-subspace E_{in} of dimension d_i, a $\mathbb{F}_2$-subspace E_{out} of dimension d_o and $d_i \leq d_o$, the computational complexity C of the limited-birthday algorithm solving **Limited-birthday**(P, E_{in}, E_{out}) satisfies:*

$$\log_2(C) = \begin{cases} (n - d_o)/2 & if\ n < 2d_i + d_o\,, \\ n - d_i - d_o & otherwise. \end{cases}$$

The optimality of this algorithm has been proven by Iwamoto et al [IPS13].

The non-random behavior that we exhibit in Section 4 is based upon Problem 1. We provide an algorithm which solves an instance of this problem with a lower complexity than the one given by Theorem 1.

Remark 3. Solving instanciations of Problem 1 is not the only existing type of distinguishing attack. For instance, Lamberger et al. [LMR$^+$09], Jean et al. [JNPP13] and Gilbert [Gil14] consider some other ones.

3.2 A 11-Round Truncated Differential Path over P_{1024}

Introduced by Knudsen in 1995 [Knu95], truncated differentials consider only whether a byte position is affected by a non-zero differential value or not, where a differential value is the difference between two state values. Plenty of hash function cryptanalysis have been analyzed from this perspective, among which [Pey07, MNPN$^+$09, MPRS09, JF11, JNPS12, JNPP14]. Figure 3 specifies the truncated differential path at the center of our attack: Blue cells denote *active* bytes, where there might be non-zero differences, and white cells denote *non-active* bytes where there are zero-differences (see Section 3.3 for an explanation on how we have obtained this path).

For such a truncated description, all transformations besides **MB** are deterministic. **MB** induces probabilistic truncated transitions: a column value δ which is non-zero on u chosen byte positions has an image $\mathbf{MB}(\delta)$ vanishing in v chosen byte positions with probability β^{-v} as long as the MDS property is satisfied, and 0 otherwise.

$$\Pr\left(\begin{array}{l}\mathbf{MB}(\delta) \text{ vanishes on } v \text{ chosen byte positions} \\ \mid \delta \text{ is non-zero on } u \text{ chosen byte positions}\end{array}\right) = \begin{cases} \beta^{-v} & \text{if } u + (8 - v) \geq 9\,, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Coding theory ensures that the same behavior applies to $\mathbf{MB}^{-1}$.

Most rebound attacks rely on truncated differential paths which are diffusing to full active state in three rounds backward and forward. The sequence of numbers of active bytes is a classical representation of such truncated differential paths. The sequence of the differential path used by the 10-round rebound attack of [JNPP14] is

$$64 \xrightarrow{R_1} 8 \xrightarrow{R_2} 1 \xrightarrow{R_3} 8 \xrightarrow{R_4} 64 \xrightarrow{R_5} 128 \xrightarrow{R_6} 64 \xrightarrow{R_7} 8 \xrightarrow{R_8} 1 \xrightarrow{R_9} 8 \xrightarrow{R_{10}} 64\,.$$

Our truncated differential path is very different, especially the sequence of numbers of active bytes outlines how structured our path is (see Figure 3),

$$104 \xrightarrow{R_1} 53 \xrightarrow{R_2} 34 \xrightarrow{R_3} 34 \xrightarrow{R_4} 34 \xrightarrow{R_5} 34 \xrightarrow{R_6} 34 \xrightarrow{R_7} 34 \xrightarrow{R_8} 34 \xrightarrow{R_9} 53 \xrightarrow{R_{10}} 104 \xrightarrow{R_{11}} 128\,.$$

A first analysis consists in evaluating the plausibility of such a truncated differential path: the probability that there exists a pair of inputs such that their difference when propagated through the permutation rounds agrees with the patterns of the truncated differential path.

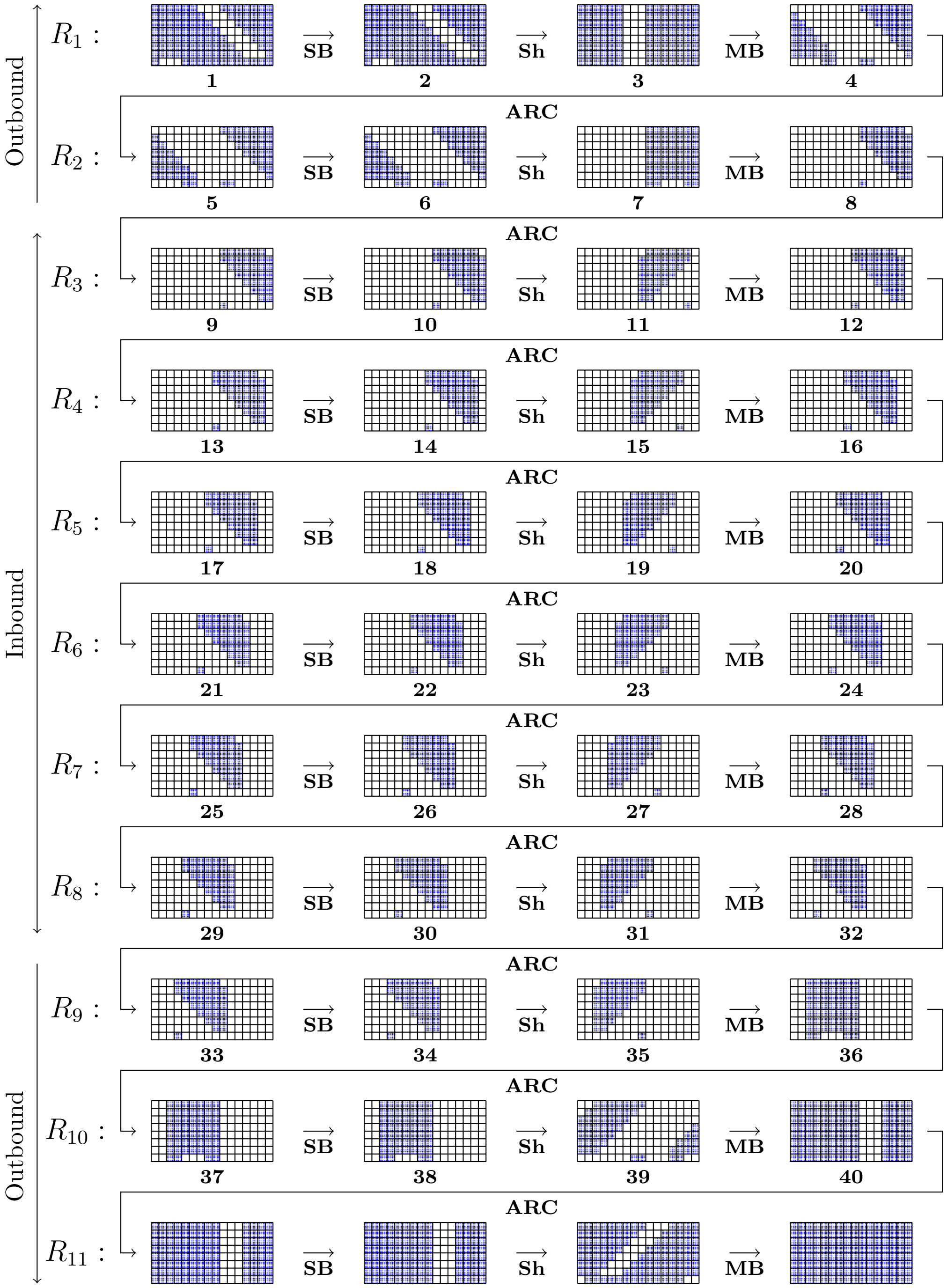


Figure 3: A 11−round truncated differential path over P_{1024} of Grøstl-512.

Starting with β^{104} differences, there exist β^{128} (ordered) pairs of internal state values that have their difference equaling each differential value, *i.e.* a total number of β^{232} pairs of input values. From Equation (3), applied independently on each column, we can estimate the diffusion probability of the differential path given in Figure 3: β^{-51} for the propagation through the Round 1 **MB** transformation, β^{-22} for Rounds 2 to 8 and β^{-3} for Round 9. Rounds 10 to 11 are deterministic. By Assumption (2), $\beta^{128+104} \cdot \beta^{-51+7\cdot(-22)-3} = \beta^{232} \cdot \beta^{-208} = \beta^{24} = 2^{192}$ input pairs shall thus fulfill our 11-round differential path.

From these considerations, we got confident in finding pairs of input values whose differences when propagated through the successive transformations agree with the patterns of the truncated differential path of Figure 3. Such a pair solves the problem **Limited-birthday**$(P_{1024}, \Delta_{in}, \Delta_{out})$ where Δ_{in} and Δ_{out} are both $\mathbb{F}_2$-subspace of differential values of dimension 104. From Theorem 1, we know that the computational complexity of the generic attack is $\beta^{(128-104)/2} = \beta^{12} = 2^{96}$. Our distinguisher is an algorithm that finds such a pair with a computational complexity lower than β^{12}.

In comparison, we have $\beta^{64} \cdot \beta^{128} = \beta^{192}$ pairs of inputs that satisfy the input differential value of the path used in [JNPP14]. Rounds 1 and 7 propagations have then probability β^{-56}, the **MB** propagation probability of Rounds 2 and 8 is β^{-7} and the one of Round 6 is β^{-64}. Since the other rounds have a deterministic behavior we have therefore $\beta^{192} \cdot \beta^{-190} = \beta^{2} = 2^{16}$ input pairs that fulfill this 10-round truncated differential path.

Note that such an analysis benefits to be made from the middle rounds. Starting at Round 6 of the path given in Figure 3, we have $\beta^{34} \cdot \beta^{128} = \beta^{162}$ pairs of values of internal state that have their difference agreeing with Pattern 21. Propagations for Round 6 to Round 11 through **MB** or for Round 6 to Round 2 through $\mathbf{MB}^{-1}$ happen both with probability $\beta^{-22} \cdot \beta^{-22} \cdot \beta^{-22} \cdot \beta^{-3} = \beta^{-69}$ and we retrieve that $\beta^{162} \cdot \beta^{-2\cdot 69} = \beta^{24}$ input pairs shall fulfill our 11-round differential path.

This analysis derives from rebound attacks. They share an overall strategy, which splits into two phases. The first step is called *inbound phase* and consists in finding several pairs of values in the middle of the truncated differential path such that this path is verified for as many rounds as possible in the middle of the path. The second step consists in enumerating these pairs to find one satisfying remaining probabilistic transitions of the truncated differential path in outward directions and is called *outbound phase.* Our inbound phase involves Rounds 3 to 8 and consists in collecting pairs of full state values whose differences agree with Pattern 8 and when propagated until Round 9 with Pattern 32. Our outbound phase consists in finding one among these pairs which satisfies the two remaining non deterministic transitions: through **MB** in Round 9 and $\mathbf{MB}^{-1}$ in Round 2. Each holds with probability β^{-3} and simultaneously with probability $\beta^{-6} = 1/2^{48}$.

3.3 Searching Sparse Truncated Differential Paths

Looking for truncated differential paths well-adapted to rebound attacks is closely related to the problem of finding low-weight differential paths in an AES-like block cipher. Mixed-Integer Linear Programming (MILP) solvers turn out be very efficient to solve such problems [MWGP11]. We explain here how the differential paths needed for the inbound and outbound phases can be found with this approach.

Inbound Rounds

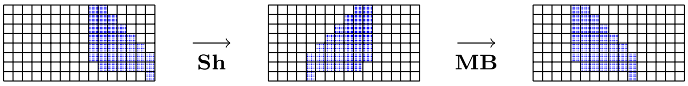
We aim at finding long differential paths that span over a as small as possible number of columns, in the hope that it yields in return many more register values that verify the path. More precisely, we look for a small number of non-zero columns in the 6 inner inbound rounds: 7 columns at each round (it is unfeasible to find a solution for 6 columns). Instead of minimizing the number of active S-Boxes in the support of the differential values as in a block-cipher context, we thus have to minimize the number of active columns.

Precisely, we define 7 × 128 decision Boolean variables x_i, each seen as one byte of the 7 registers of the permutation P_{1024} restricted to 6 rounds: the input, the 5 internal and the output registers. These variables encode truncated differential paths: the variable x_i equal to 0 or 1 depending on whether or not a differential path is active at this byte. As it is now classical, we can write the action of **Sh** and **MB** as 6×16 linear inequalities between the x_i's of the type

$$\sum_{x \in \mathcal{P}} x + \sum_{x' \in \mathcal{P}'} x' \geq 9\, t \text{ and } \ \forall\, v \in \mathcal{P} \cup \mathcal{P}',\ t \geq v\,.$$

The sets $\mathcal{P}$ (resp. $\mathcal{P}'$) are subsets with 8 variables that span the r-th round register (resp. the $r+1$-th round register) and the variables t that indicate whether a column is active or not. We furthermore add 6 inequalities, which state that the sum of the 16 variables t defined by Round r is at most 7. This done, we ask for minimizing the sum of these 6×16 variables t.

After few minutes with the GUROBI solver [Gur17], we found numerous such paths. Most of them are trivially linked: any shift by a fixed amount of a differential path yields an equivalent path. Finally, two iterative paths catch our attention, the one that we use (see Figure 3) and a second one defined as follows (that does not seem to be better for our purpose).



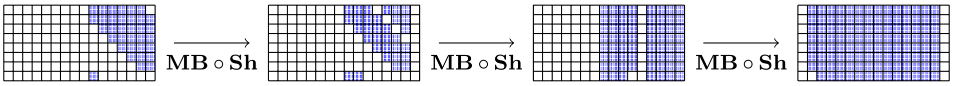
Note that in both cases, five columns do not reach the MDS bound and we can easily derive from them sparser paths (see Remark 5 in Section 4.4).

Remark 4. The number of non-zero columns depends directly on **Sh**. But more obvious choices lead to much worse behaviors, for instance shifting by 0, 1, 2, 3, 4, 5, 6 and 7 (instead of 11) positions yields a reproducible differential pattern on only 5 columns.

Outbound Rounds

We take advantage of this MILP approach for searching low-cost outbound differential characteristic as well. For this task, we define the 128 first variables $x_0, \ldots, x_{127}$ as being equal to the "inbound" differential pattern that we have selected, we also add that after three rounds the output register must have more than 16 non-active bytes and we trace in new variables c the number of zero bytes x_i in the output in each active column by the **MB** transformation. We ask then for minimizing the sum of these variables c, under the condition than the bytes of a register can not be all equal to one.

The best path that we find is the one of Figure 3. Its output register contains 24 zero bytes, up to a transition probability equal to $\beta^{-25} = 1/2^{200}$: β^{-22} for Round 1 and β^{-3} for Round 2. We found another path in this way, but with a slightly smaller probability transition, β^{-26}, and 17 zero bytes in the output. We give it below for the sake of completeness.



3.4 SuperSBox Description

For truncated differential paths, **ARC** transformations may be ignored as they have no impact on truncated differences. Since **Sh** transformations only move the difference positions and **SB** transformations apply independently on bytes, they commute and we may permute the applications of these transformations.

Following [DR06], we now define the two following transformations.

- A non-linear **SuperSBox** (**SSB**) transformation which applies independently on columns,
$$\textbf{SuperSBox} := \textbf{SubBytes} \circ \textbf{MixBytes} \circ \textbf{SubBytes}\,.$$
We will refer as **SuperSBox** (**SSB**) indifferently to consider this transformation on full states or on single columns.

- A linear **SuperLinear** (**SL**) transformation,
$$\textbf{SuperLinear} := \textbf{ShiftBytesWide} \circ \textbf{MixBytes} \circ \textbf{ShiftBytesWide}\,.$$

Our 11-round truncated differential path given in Figure 3 may then be rewritten in a more compact form as in Figure 4.

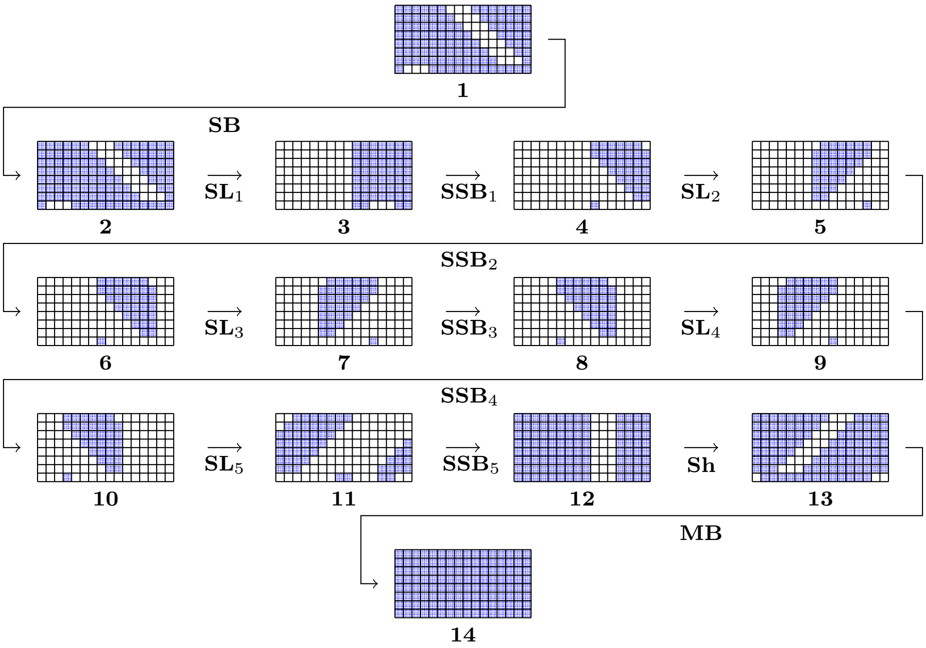


Figure 4: The 11-round truncated differential path with a SuperSBox description.

4 The Distinguisher

4.1 Notations

Byte values are seen as elements in $\mathcal{B} = \mathbb{F}_{2^8}$. Column values are seen as elements in $\mathcal{C} = \mathcal{B}^8$ and for $X \in \mathcal{C}$, we denote by X_i the i^{th} coordinate of X which is also the i^{th} byte of the column. State values are seen indifferently as elements in $\mathcal{S} \simeq \mathcal{B}^{128} \simeq \mathcal{C}^{16}$ and for $Y \in \mathcal{S}$, we denote by $Y_{i,j}$ the byte value in the i^{th} row and the j^{th} column. For now on, all References 1-14 to (grid) patterns are linked to Figure 4. For $i \in \{1, \ldots, 14\}$, P_i denotes the $\mathcal{B}$-linear subspace of differential values which agree with Pattern i (whose non-zero byte positions are included in hatched cell positions of Pattern i) and I_i is the set of column index which have active bytes. We denote by $\delta_{|j}$ the restriction of a differential value δ to the j^{th} column and by extension $(P_i)_j$ denotes the linear subspace of $\mathcal{C}$ which agree with the j^{th} column of Pattern i. We call *completion* of a partial state value V (typically a column value) a full state value whose restriction to byte positions of V is V. By extension, we call completion of a pair of partial state values defined on the same byte positions (V_1, V_2) a pair of completions of the V_i's such that their difference in other byte positions than byte positions of the V_i's is zero.

4.2 Sketch of the Algorithm

We give an overview of the distinguishing algorithm here. Detailed explanations are postponed to subsequent sections, exception made for the outbound phase, which is trivial.

Step 1. We construct a basis for Δ_1, the 12-dimensional $\mathcal{B}$-vector subspace of elements δ_1 in $\mathcal{S}$ whose non-zero byte positions are included in Pattern 4 and whose images by $\mathbf{SL}_2$, δ_1', have their non-zero byte positions included in Pattern 5:

$$\Delta_1 = \{\delta_1 \in P_4 \mid \delta_1' = \mathbf{SL}_2(\delta_1) \in P_5\}\,. \tag{4}$$

We construct a basis for $\Delta_2 = \{\delta_2 \in P_6 \mid \delta_2' = \mathbf{SL}_3(\delta_2) \in P_7\}$ and a basis for $\Delta_3 = \{\delta_3 \in P_8 \mid \delta_3' = \mathbf{SL}_4(\delta_3) \in P_9\}$ too. The use of these basis allows us to enumerate elements in these subspaces with negligible computational and memory complexities. The construction of these basis requires $O(1)$ computational and memory complexities.

Step 2. We choose an arbitrary δ_2 in Δ_2. Column by column, for all columns indexed by I_6, we store in lists $(C_i)_{i \in I_6}$ all pairs of column values whose differences are compatible with δ_2 and whose images by $\mathbf{SSB}_2^{-1}$ have differences suitable with Pattern 5. Identically for all columns indexed by I_7, we store in lists $(C_j')_{j \in I_7}$ all pairs of column values whose differences are compatible with δ_2' and whose images by $\mathbf{SSB}_3$ have differences suitable with Pattern 8. We compute thus for all i in I_6, respectively for all j in I_7, the lists C_i, respectively C_j', defined by:

$$\begin{aligned} C_i &= \{\, (X, Y) \in \mathcal{C}^2 \mid X \oplus Y = (\delta_2)_{|i}, \quad \mathbf{SSB}_2^{-1}(X) \oplus \mathbf{SSB}_2^{-1}(Y) &\in (P_5)_{|i} \,\}\,, \\ C_j' &= \{\, (X, Y) \in \mathcal{C}^2 \mid X \oplus Y = (\delta_2')_{|j}, \quad \mathbf{SSB}_3(X) \oplus \mathbf{SSB}_3(Y) &\in (P_8)_{|j} \,\}\,. \end{aligned}$$

The construction of these lists requires $O(\beta^7)$ computations and memory space.

Step 3. From $O(\beta^6)$ elements δ_1 of Δ_1, we compute and store β^6 pairs of 7-column values in a list E, built from combinations of elements in the lists $(C_i)_{i \in I_6}$, such that any completion induced by this pair of indexed 7-column values has a difference which is equal to δ_2 and has the difference of its images by $\mathbf{SL}_2^{-1} \circ \mathbf{SSB}_2^{-1}$ that lies in Δ_1. From a similar enumeration in Δ_3, in a list F, we store β^6 pairs of 7-column values built from combinations of elements in the lists $(C_j')_{j \in I_7}$ such that any completion induced by these pairs of indexed 7-column values has a difference which is equal to δ_2' and has the difference of its images by $\mathbf{SSB}_3$ lying in Δ_3. This step costs $O(\beta^6)$ computations and the same in memory.

Step 4. We find $(e, e \oplus (\delta_2)_{|I_6})$ in E and $(f, f \oplus (\delta_2')_{|I_7})$ in F such that $(f, f \oplus (\delta_2')_{|I_7})$ admits completions whose images by $\mathbf{SL}_3^{-1}$ do not contradict with $(e, e \oplus (\delta_2)_{|I_6})$. An arbitrary choice in $E \times F$ yields such completions only with probability β^{-12}. However, β^{28} completions are available whenever it does. Such a pair is found in $O(\beta^7)$ computations and $O(\beta^6)$ in memory.

Step 5. We determine a pair $(s, s \oplus \delta_3')$ of 34-byte values[1] whose byte positions are induced by Pattern 9 and whose difference is suitable with δ_3' induced by $(f, f \oplus (\delta_2')_{I_7})$. This pair is built in such a way that the image of any completion of these 34-byte values by $\mathbf{SSB}_4$ has its difference in P_{10} and that there exist β^{72} completions of $(f, f \oplus (\delta_2')_{|I_7})$ whose images by $\mathbf{SL}_4 \circ \mathbf{SSB}_3$ do not contradict with these 34-byte values. This pair of 34-byte values can be found in $O(\beta^3)$ computations.

Step 6. Among the β^{28} completions found in Step 4 and the β^{72} completions found in Step 5, we compute an intersection of β^6 completions. Propagated outwards, all these β^6 pairs of full state values have their differences following the truncated differential path from Pattern 4 to Pattern 10. This has $O(\beta^9)$ computational complexity, requires $O(\beta^7)$ memory space and concludes the inbound phase.

Step 7. By enumerating the β^6 pairs of full state values collected in Step 6, we find a pair which satisfies simultaneously both remaining independent probabilistic transitions: through **MB** from Pattern 10 to Pattern 11 and through $\mathbf{MB}^{-1}$ from Pattern 4 to Pattern 3. This outbound phase requires $O(\beta^6)$ computations.

To summarize, this algorithm constructs a pair of full state values such that when propagated through 11 rounds of P_{1024}, the successive differences agree with the truncated differential path of Figure 4. This is done with $O(\beta^9) \simeq 2^{72} < 2^{80}$ computational complexity and $O(\beta^7) \simeq 2^{56} < 2^{64}$ memory complexity. The generic attack on such input and output patterns requires about $\beta^{12} = 2^{96}$ computations (see Section 3.1).

4.3 Step 1: Construction of the Basis of Linear Subspaces Δ_i

We explain how to construct a basis of Δ_1, the $\mathcal{B}$-vector space defined by Equation (4). Making explicit the $\mathbf{SL}_2$ transformation yields Figure 5.

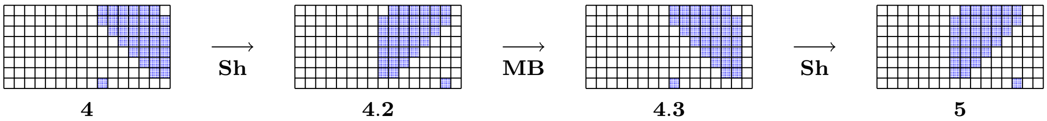


Figure 5: The truncated differential path verified by elements of the set Δ_1.

From the 34 hatched cells of Pattern 4 , we see that P_4 is a $\mathcal{B}$-linear subspace of dimension 34. Recalling that **MB** applies independently on columns, we first focus on the transition of the 9^{th} column of Pattern 4.2 through **MB** (see Figure 6).

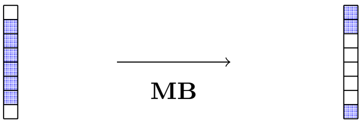


Figure 6: Differential through a **MixBytes** transformation.

Since **MB** satisfies the MDS property, the following subspace is of dimension 1:

$$\{ X \in \mathcal{C} \mid X_1 = X_8 = 0 \text{ and } (\mathbf{MB}(X))_3 = \ldots = (\mathbf{MB}(X))_7 = 0 \} .$$

By Gaussian elimination, we find $\{b_9\}$, a basis of this subspace. The same procedure yields $\{b_{15}\}$, a basis of the subspace of dimension 1 corresponding to Column 15 and

[1] By a slight abuse of notation, δ_3' refers here indifferently to differential values that span the whose state or to the restriction to the active bytes.

$\{b_{10}, b'_{10}\}, \ldots, \{b_{14}, b'_{14}\}$ basis of the subspaces of dimension 2 corresponding respectively to Columns 10 to 14. We construct then full state values $(Y_i)_{i \in I_4}$ and $(Y'_i)_{i \in \{10,\ldots,14\}}$ as completions of $(b_i)_{i \in I_4}$ and $(b'_i)_{i \in \{10,\ldots,14\}}$ with 0 bytes in the remaining bytes positions. A basis of Δ_1 is then given by

$$\left\{ \mathbf{Sh}^{-1}(Y_9),\, \mathbf{Sh}^{-1}(Y_{15}),\, \mathbf{Sh}^{-1}(Y_{10}),\, \mathbf{Sh}^{-1}(Y'_{10}),\, \ldots,\, \mathbf{Sh}^{-1}(Y_{14}),\, \mathbf{Sh}^{-1}(Y'_{14}) \right\} .$$

Basis for Δ_2 and Δ_3 are built in a same way.

4.4 Step 2: Pairs of Columns with Input/Output SuperSBox Differences

Let δ_2 be an element of Δ_2. The purpose of this step is to compute and store the lists C_i for all $i \in I_6$ and the lists C'_j for all $j \in I_7$. Exhaustive search could achieve this goal in $14 \cdot \beta^8$ computations but we will show now how to do it faster, in the spirit of [SLW+10].

Let us focus on C_8, that stores elements in $\mathcal{C}^2$ whose difference equals $(\delta_2)_{|8}$ and whose images by $\mathbf{SSB}_2^{-1}$ have their difference in $(P_5)_{|8}$. To this purpose, Figure 7 introduces two intermediate patterns by decomposing $\mathbf{SSB}_2$ transformation applied on the arbitrarily chosen 8^{th} column.

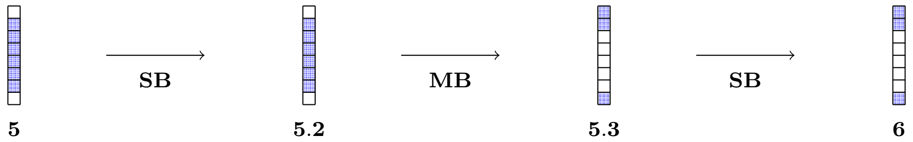


Figure 7: A truncated differential through a SuperSBox operation.

We denote by D_1 the set of differential values agreeing with Pattern 5.3 of Figure 7 and having their image by $\mathbf{MB}^{-1}$ transformation agreeing with Pattern 5.2,

$$D_1 = \{\delta \in (P_6)_{|8} \mid \mathbf{MB}^{-1}(\delta) \in (P_5)_{|8}\} .$$

From Equation (3), we have $|D_1| = \beta$. We now denote by D_2 the set

$$D_2 = \{\delta \in \mathcal{C} \mid \exists\, X \in \mathcal{C} \text{ s.t. } \mathbf{SB}^{-1}(X \oplus (\delta_2)_{|8}) \oplus \mathbf{SB}^{-1}(X) = \delta\} .$$

From Equation (1), $|D_2| = \beta^3 \cdot 2^{-3}$ and

$$\delta \in D_2 \Rightarrow \left|\{X \in \mathcal{C} \mid \mathbf{SB}^{-1}(X \oplus (\delta_2)_{|8}) \oplus \mathbf{SB}^{-1}(X) = \delta\}\right| = 2^3 \cdot \beta^5 .$$

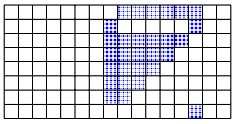
By Assumption (2), $|D_1 \cap D_2| = 2^{-3} \cdot \beta$. The following list has then cardinal β^6:

$$C_8 = \bigcup_{\delta \in D_1 \cap D_2} \{(X, X + (\delta_2)_{|8}) \in \mathcal{C}^2 \mid \mathbf{SB}^{-1}(X \oplus (\delta_2)_{|8}) \oplus \mathbf{SB}^{-1}(X) = \delta\} .$$

To compute C_8, we consider all pairs $(X, X \oplus (\delta_2)_{|8}) \in \mathcal{C}^2$ such that the restriction of X on the non-active bytes (white cells) is 0. We store, in an intermediate list H, pairs verifying that the restriction of $\mathbf{MB}^{-1}(\mathbf{SB}^{-1}(X \oplus (\delta_2)_{|8}) \oplus \mathbf{SB}^{-1}(X))$ to the first and eighth byte positions is 0. We store then in C_8, ordered according to the sorting key $\delta(X) = \mathbf{SSB}_2^{-1}(X) \oplus \mathbf{SSB}_2^{-1}(X \oplus (\delta_2)_{|8})$, all elements in $\mathcal{C}^2$ such that their restriction to the first, second and eight byte positions equals restriction to those byte positions of some element in H and such that on the remaining byte positions, the difference is 0. For now on, when considering elements in C_8 it could be following the context initial pairs of values or their images through $\mathbf{SSB}_2^{-1}$.

Remaining lists are computed with the same routine, the computational complexity to construct them correspond to their memory complexity, *i.e.* the size of the lists: C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14} are respectively of size β^6, β^7, β^6, β^5, β^4, β^3, β^3 and C'_6, C'_7, C'_8, C'_9, C'_{10}, C'_{11}, C'_{12} are respectively of size β^3, β^3, β^4, β^5, β^6, β^7, β^6.

Remark 5. We could have considered another truncated differential path, sparser, which is strictly included in the path of Figure 3, replacing Pattern 5 with the following Pattern 5′.



5′

We expect this path to be realized by fewer pairs of full state values. The selection of this pattern reduces the dimension of Δ_1 from 12 to 7. It remains large enough to mount the attack and induces here lists C_i and C'_i of maximum size β^6. This step has then $O(\beta^6)$ computational and memory complexities.

4.5 Step 3: Pairs of Partial States Satisfying Δ_1 to δ_2 and δ_2 to Δ_3

We pick[2] arbitrary differential values δ_1 in Δ_1. For each δ_1 in Δ_1, we compute its image δ'_1 by $\mathbf{SL}_2$. For i in I_6, we consider all pairs of columns values in C_i computed in Step 2 whose difference through $\mathbf{SSB}_2^{-1}$ equals $(\delta'_1)_{|i}$. The computational cost is simply a search in a sorted list. Whenever a match is found simultaneously for each of these seven columns, we store in a list E the pairs of 7-column values computed as the concatenations of all corresponding pairs of columns values in the C_i, whose difference is $(\delta_2)_{|I_6}$ and whose images through $\mathbf{SSB}_2^{-1}$ have a difference equal to $(\delta'_1)_{|I_6}$. We reproduce this routine until we get β^6 such pairs of 7-column values. This requires to enumerate $O(\beta^6)$ elements in Δ_1 (see Remark 6). This done, we get the desired list E with β^6 elements.

A list F of β^6 pairs of 7-column values whose differences equal δ'_2 and whose images by $\mathbf{SSB}_3$ have a difference which equals $(\delta_3)_{|I_7}$ for some δ_3 in Δ_3 is built following the same procedure with the lists C'_j.

Remark 6. By Assumption (1), the map $\mathbf{SSB}_2^{-1}$ has the behavior of an ideal SBox applied independently on each column. For a fixed δ'_1, whenever there exists X in $\mathcal{C}^7$ such that

$$\mathbf{SBB}_2^{-1}(X \oplus (\delta_2)_{|I_6}) \oplus \mathbf{SBB}_2^{-1}(X) = (\delta'_1)_{|I_6}),$$

which holds with probability 2^{-7}, we have 2^7 elements X with the same property.

Since $C_8, \ldots, C_{14}$ computed in Step 2 store all possible pairs of column values whose differences equal respectively $(\delta_2)_{|8}, \ldots, (\delta_2)_{|14}$ and whose images have a difference lying in P_5, we find from the lists C_i, with a probability of 2^{-7} for an arbitrary element δ_1 in Δ_1, 2^7 pairs of partial state values, on columns indexed by I_6, whose differences equal $(\delta_2)_{|I_6}$ and images by $\mathbf{SSB}_2^{-1}$ have a difference which equals $(\delta'_1)_{|I_6}$.

4.6 Step 4: First Patching up

We want to find $(e, e \oplus (\delta_2)_{|I_6})$ in E and $(f, f \oplus (\delta'_2)_{|I_7})$ in F such that there exists a completion of $(f, f \oplus (\delta'_2)_{|I_7})$ whose image by $\mathbf{SL}_3^{-1}$ does not contradict with $(e, e \oplus (\delta_2)_{|I_6})$. We call such a completion a *matching completion* of $(f, f \oplus (\delta'_2)_{|I_7})$ with $(e, e \oplus (\delta_2)_{|I_6})$. Since $\delta'_2 = \mathbf{SL}_3(\delta_2)$, we know that any matching completion of f with e is a matching completion of $f \oplus (\delta'_2)_{|I_7}$ with $e \oplus (\delta_2)_{|I_6}$. We show now that for an arbitrary

[2] We make use of the basis computed at Step 1 for the 12-dimensional $\mathcal{B}$-vector subspace Δ_1.

$\big((e, e \oplus (\delta_2)_{|I_6}),\, (f, f \oplus (\delta_2')_{|I_7})\big)$ in $E \times F$, a matching completion exists only with probability β^{-12}. We then show how to find such a pair in $O(\beta^7)$ computations and $O(\beta^6)$ in memory.

Figure 8 introduces two intermediate patterns by decomposing the $\mathbf{SL}_3$ transformation. Pink cells in Pattern 6.3 correspond to byte values fixed by f whereas green cells in Pattern 6.2 correspond to byte values fixed by e.

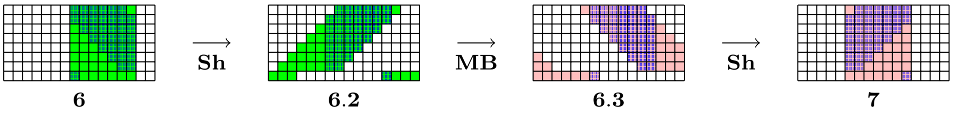


Figure 8: Fitting the two pieces, first edition.

MB applies independently on columns, we can therefore analyze local columns transitions. Two types of columns have to be distinguished: In columns 7 to 14, there are too many of the c constraints (green cells in Pattern 6.2) to be compensated by the d degrees of freedom (white cells in 6.3). A local matching completion is then possible only with probability β^{d-c}. In the remaining columns, there are enough of the d degrees of freedom (white cells in Pattern 6.3) to compensate the c constraints (green cells in Pattern 6.2). We have β^{d-c} local matching completions for each of these columns.

We do now the exact analysis for the 7^{th} column. Let's denote by $M = \{m_{i,j}\}$ the matrix of **MB**. Two degrees of freedom are available (byte positions 1 and 8) and three constraints are imposed by fixed values (byte positions 1, 2 and 8). Here, we want to determine given values $x_2, \ldots, x_7$ and y_1, y_2, y_8 whether there exist x_1 and x_8 verifying Equation (5) or not,

$$\forall \ell \in \{1, 2, 8\},\ \sum_{j=2}^{7} m_{\ell,j} x_j + m_{\ell,1} x_1 + m_{\ell,8} x_8 = y_\ell\,. \tag{5}$$

Since M is MDS, the minor $m_{1,1} \cdot m_{2,8} - m_{1,8} \cdot m_{2,1}$ is non zero, we can then write

$$x_1 = l_{1,7}(x_2, \ldots, x_7) + g_{1,7}(y_1, y_2) \quad \text{and} \quad x_8 = l_{8,7}(x_2, \ldots, x_7) + g_{8,7}(y_1, y_2)\,,$$

where $g_{1,7}$, $g_{8,7}$, $l_{1,7}$ and $l_{8,7}$ are linear forms depending only on **MB** and on the positions of the fixed bytes. Introducing L_7 and R_7 two linear forms obtained by substituting x_1 and x_8 by their expressions, Equation (5) is then equivalent to

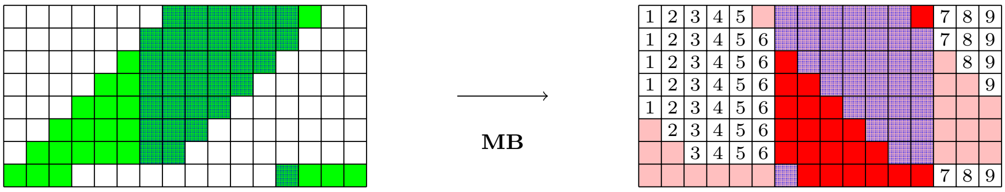
$$\begin{cases} x_1 = l_{1,7}(x_2, \ldots, x_7) + g_{1,7}(y_1, y_2)\,, \\ x_8 = l_{8,7}(x_2, \ldots, x_7) + g_{8,7}(y_1, y_2)\,, \\ L_7(x_2, \ldots, x_7) = R_7(y_1, y_2, y_8)\,. \end{cases}$$

A match between e and f is then possible on the 7^{th} column if and only if $L_7(e_{2,7}, \ldots, e_{7,7})$ equals $R_7(f_{2,1}, f_{2,2}, f_{2,8})$. For the sake of simplicity, we drop the byte positions and shall rather write $L_7(e)$ and $R_7(f)$.

We find the two linear forms L_{13} and R_{13} by conducting the same analysis for the 13^{th} column. For $i \in \{8, \ldots, 12\}$, we find couple of linear forms L_i, L_i' and R_i, R_i'. We get indeed two pairs of linear forms since the difference between the number of constraints and the degree of freedom is not 1 anymore but 2. We reproduce these arguments for the remaining Columns 1 to 6 and 14 to 16. As there are locally more degrees of freedom than constraints, each choice of $\big((e, e \oplus (\delta_2)_{|I_6}),\, (f, f \oplus (\delta_2')_{|I_7})\big)$ in $E \times F$ admits local matching completions. Finally, we have a matching completion of f with e if and only if the twelve equations $L_7(e) = R_7(f)$, $L_{13}(e) = R_{13}(f)$, $L_8(e) = R_8(f)$, $L_8'(e) = R_8'(f)$,

…, $L_{12}(e) = R_{12}(f)$, $L'_{12}(e) = R'_{12}(f)$ are simultaneously satisfied. Uniformly distributed values fulfill this set of equations with probability β^{-12}.

We show now how to find a pair $\big((e, e \oplus (\delta_2)_{|I_6}),\ (f, f \oplus (\delta'_2)_{|I_7})\big)$ in $E \times F$ for which a matching completion of f with e exists in $O(\beta^7)$ computations: We sort the list F according to the lexicographic order given by $(R_7(f), R_{13}(f), R_8(f), R'_8(f), \ldots, R_{12}(f), R'_{12}(f))$ in $O(\beta^7)$ computations and $O(\beta^6)$ in memory. For each element $(e, e \oplus (\delta_2)_{|I_6})$ of E, we compute the value $(L_7(e), L_{13}(e), L_8(e), L'_8(e), \ldots, L_{12}(e), L'_{12}(e))$ and we check if it belongs to the previous sorted list. This costs $O(\beta^6)$ computations and $O(\beta^6)$ searches in a sorted list. Since β^{12} pairs $\big((e, e \oplus (\delta_2)_{|I_6}),\ (f, f \oplus (\delta'_2)_{|I_7})\big)$ in $E \times F$ are available and since a matching completion of f with e exists with probability β^{-12}, we expect to find a match. No pair admitting a matching completion should happen, we start again from Step 2 with another δ_2. For now on, we suppose that such a pair exists. We find it in $O(\beta^7)$ computations and $O(\beta^6)$ in memory.



6.2 **6.3**

Figure 9: Fixed bytes and lists of elements.

The pair $\big((e, e \oplus (\delta_2)_{|I_6}),\ (f, f \oplus (\delta'_2)_{|I_7})\big)$ in $E \times F$ being now fixed, all byte values in Columns 7 to 13 of Pattern 6.3 are fixed by this choice (red cells in Figure 9).

We compute now all matching completions of f with e. We construct for each of the nine remaining columns the lists of column values satisfying the linear constraints imposed by fixed bytes. The construction of these lists does not differ from what we just did with the use of linear forms analogous to $g_{i,j}$ and $f_{i,j}$. We have then, as illustrated by Figure 9, B_1, B_2, B_3, B_4 and B_9, lists of β^4 elements in the 1^{st}, 2^{nd}, 3^{rd}, 4^{th} and 16^{th} columns, B_5 and B_8, lists of β^3 elements in the 5^{th} and the 15^{th} columns and B_6 and B_7, lists of β elements in the 6^{th} and 14^{th} columns. We get then β^{28} possible matching completions as direct product of the lists B_i (we cannot store this product in a unique list, since this would require outrageous computational and memory complexities).

4.7 Step 5: Second Patching up

At this point of the algorithm, the differential value δ_2 has been fixed. We have built E, a list of β^6 pairs of 7-column values indexed by I_6 such that their difference is $(\delta_2)_{|I_6}$ and the difference of the images of any completion by $\mathbf{SL}_2^{-1} \circ \mathbf{SSB}_2^{-1}$ is in Δ_1. Similarly, we have built F, a list of β^6 pairs of 7-column values indexed by I_7 such that their difference is $(\delta'_2)_{|I_7}$ and the difference of the images of any completion by $\mathbf{SSB}_3$ is in Δ_3. We have found $(e, e \oplus (\delta_2)_{|I_6})$ in E and $(f, f \oplus (\delta'_2)_{|I_7})$ in F such that there exist β^{28} matching completions of f with e, whose images by $\mathbf{SL}_3^{-1}$ do not contradict with e. Columns e and f are then now fixed. We call $\delta_1 \in \Delta_1$ and δ_3 in Δ_3 the differential values induced by these choices when propagating $(e, e \oplus (\delta_2)_{|I_6})$ backward by $\mathbf{SSB}_2^{-1}$ and $(f, f \oplus (\delta'_2)_{|I_7})$ forward by $\mathbf{SSB}_3$. We denote by $(f', f' \oplus (\delta_3)_{|I_7})$ the image of $(f, f \oplus (\delta'_2)_{|I_7})$ by $\mathbf{SSB}_3$.

Each of these β^{28} pairs of full state values, when propagated backward and forward, have their differences agreeing with Figure 4 from Pattern 4 to 9. To fulfill the whole truncated differential path, three independent probabilistic transitions remain: through

$\mathbf{SSB}_1^{-1}$ verified with probability β^{-3}, through $\mathbf{SSB}_4$ verified with probability β^{-22} and through $\mathbf{SL}_5$ verified with probability β^{-3}. Enumerating the β^{28} pairs of full state values, we expect to find one satisfying simultaneously these three independent probabilistic transitions. However, to be a distinguisher, our algorithm has to find such a pair in less than β^{12} computations.

Independently of the choice of $(e, e \oplus (\delta_2)_{|I_6})$ in E and focusing on $(f', f' \oplus (\delta_3)_{|I_7})$ induced by the choice of $(f, f \oplus (\delta'_2)_{|I_7})$ in F, we show that by fixing new byte values, we may avoid the probabilistic transition through $\mathbf{SSB}_4$. We denote by δ'_3 the image $\mathbf{SL}_4(\delta_3) \in P_9$. We look for pairs of state values $(s, s \oplus \delta'_3)$ such that $\mathbf{SSB_4}(s) \oplus \mathbf{SSB_4}(s \oplus \delta'_3) = \delta_4$ is in P_{10}. As we are not interested in the exact value of δ_4, we only need to know whether $(\mathbf{MB} \circ \mathbf{SB})(s) \oplus (\mathbf{MB} \circ \mathbf{SB})(s \oplus \delta'_3)$ is in P_{10}. The 34 bytes of s corresponding to the positions of active bytes (hatched cells in Pattern 9 of Figure 4) suffice to determine whether δ_4 is in P_{10}.

A column by column study has to be done which is more or less the same as the analysis made in Step 2. By reconsidering Figure 7, which depicts the behavior of Column 4, we see that knowing the values of 6 bytes of $s_{|4}$ is enough to determine the truncated behavior of their difference. As differential values on this column have to vanish in 5 bytes position after a $\mathbf{MB}$ transformation, we compute and store in a list L_1 the β pairs of 6-byte values: whose differences equal $(\delta'_3)_{|4}$ restricted to active byte positions and whose image by $\mathbf{SSB}_4$ of any completion with zero difference on the remaining byte positions have their differences which lie in P_{10}.

We similarly construct the six lists associated with the six remaining columns to get β pairs of 6-byte values in a list L_1 (Column 4), β^2 pairs of 7-byte values in a list L_2 (Column 5), β^2 pairs of 6-byte values in a list L_3 (Column 6), β^2 pairs of 5-byte values in a list L_4 (Column 7), β^2 pairs of 4-byte values in a list L_5 (Column 8), β^2 pairs of 3-byte values in a list L_6 (Column 9) and β pairs of 3-byte values in a list L_7 (Column 10). Corresponding to all possible combinations of pairs of byte values in the lists L_i, we get then β^{12} pairs of 34-byte values whose differences are suitable with δ'_3 and such that the images by $\mathbf{SSB}_4$ of any completion with zero difference in the remaining byte positions lie in P_{10}.

Now we determine when such choices can be patched up with f. We focus here deliberately on f without considering e. Patching constraints imposed by the choice of e and the choice of 34-byte values is considered in Step 6. Figure 10 introduces two intermediate patterns by decomposing $\mathbf{SL}_4$ transformation. On Pattern 9, yellow cells indexed by a number i correspond to positions where bytes are fixed by a choice of an element the list L_i and so does its shifted version in Pattern 8.3. On Pattern 8, pink cells correspond to the image of f by $\mathbf{SSB}_3$ and so does its shifted version in Pattern 8.2.

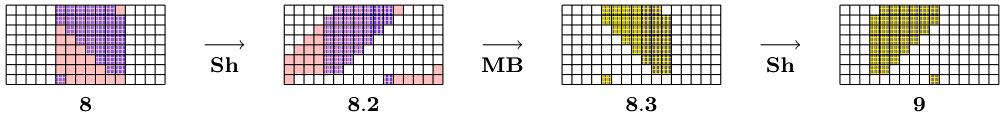
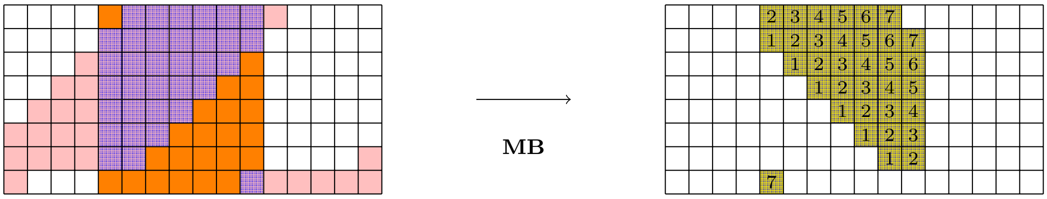


Figure 10: Fitting the two pieces, second edition.

We can reproduce the column by column analysis made in Step 4: For Column 5, there are too many of the 3 constraints (yellow cells in Pattern 8.3) to be compensated by the 2 degrees of freedom (white cells in Pattern 8.2) a local matching completion is possible only with probability β^{-1}. Columns 6 to 11 behave as Column 5: a local matching completion is possible only with probability β^{-1}. By Assumption (2), these seven independent events occurs simultaneously with probability β^{-12}. We expect then to find one among our β^{12} possible combinations of pairs in the lists L_i such that there exists a global matching completion which ensures this transition. We could proceed to a naive greedy enumeration, however, inspired by [JNPP14], we give now a procedure to find these fitting pairs of 34-byte values in $O(\beta^3)$ computations.



8.2 **8.3**

Figure 11: Merging lists, first edition.

We focus on the 5^{th} column of Pattern 8.3. A triplet (l_1, l_2, l_7) in $L_1 \times L_2 \times L_7$ admits a local matching completion with 5^{th} column of Pattern 8.2 if and only if it satisfies an affine equation (5 degrees of freedom and 6 constraints). Enumerating elements (l_1, l_2) in $L_1 \times L_2$, we compute the value it induces for l_7 (yellow cell in 8^{th} byte of 5^{th} column that satisfies the affine equation). By Assumption (2), we have a unique element in L_7 verifying this property and we store the triplet (l_1, l_2, l_7) in a list L'_1. In $O(\beta^3)$. We collect then β^3 such triplets.

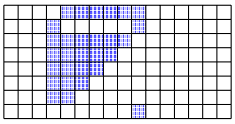
We focus now on the 6^{th} column of Pattern 8.3. A triplet (l_1, l_2, l_3) in $L_1 \times L_2 \times L_3$ admits a local matching completion with 6^{th} column of Pattern 8.2 if and only if (l_1, l_2, l_3) satisfies two affine equations (5 degrees of freedom and 7 constraints). By Gaussian elimination, we get an affine equation in l_1 and l_2 and not l_3. We use this equation to filter L'_1 to get a list L''_1 with β^2 elements suitable with this equation. Enumerating elements in L''_1, we compute, using the second equation, the value it induces for l_3 (yellow cell in 1^{st} byte of 6^{th} column that satisfies the affine equation). By Assumption (2), for each triplet (l_1, l_2, l_7) in L''_1, we search the β elements in L_3 verifying this property and store all suitable quartets (l_1, l_2, l_3, l_7) in a list L'_2. In $O(\beta^3)$, we collect then β^3 such quartets.

Reproducing this routine, filtering then expanding, for Columns 7, 8 and 9, we get a list of 7-uplet L'_3 with β^3 elements admitting local matching completions for Columns 5 to 9. This is done with $O(\beta^3)$ computations.

An element of L'_3 admits a local matching completion in Column 10 if it satisfies two affine equations (1 degree of freedom and 3 constraints). We use these equations to filter L'_3 to get a list L'_4 of β elements admitting a local matching completion in Columns 10. An element of L'_4 admits a local matching completion in Column 11 if it satisfies an affine equation (2 degrees of freedom and 3 constraints). We use these equations to filter L'_4 to get a unique 7-uplet admitting local matching completions on all the 5^{th} to 11^{th} columns.

Once a combination of elements in the lists L_i admitting local matching completions on all 5^{th} to 11^{th} columns is found, which requires $O(\beta^3)$ computations, a single local completion on all these seven columns exists. This means that the corresponding pairs of values of the 22 remaining bytes of Columns 5 to 11 are thereby fixed (orange cells of Pattern 8.2 of Figure 11). Since no constraint limits possible completions in the remaining columns, there are then β^{72} completions of $(f, f \oplus (\delta'_2)_{|I_7})$ suitable with the choice $(s, s \oplus \delta_3)$.

Remark 7. As we already noticed in Remark 5, we could have considered here also another truncated differential path, sparser and strictly included in the path of Figure 3, replacing Pattern 9 with the following pattern.



9′

To select this pattern reduces the dimension of Δ_3 from 12 to 7. It remains large enough to mount the attack. Such a pattern implies the existence of β^7 pairs of 29-byte values suiting the choice of $(f, f \oplus (\delta'_2)_{|I_7})$ with probability β^{-7}. The merging algorithm may then be done faster, in $O(\beta^2)$ computations.

4.8 Step 6: Third Patching up

Among the β^{28} matching completions found in Step 4 and the β^{72} matching completions found in Step 5, we compute and store the intersection of β^6 completions that verify the 22 constraints induced by the map **MB**. Figure 12 introduces two intermediate patterns to help clarify how to determine this intersection.

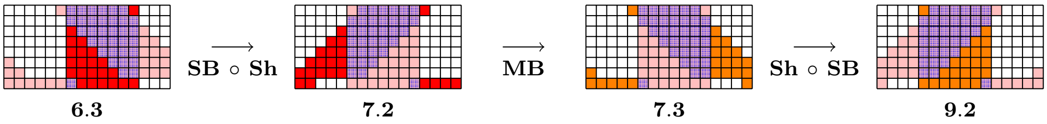


Figure 12: Fitting the two pieces, third edition.

Merging techniques of [JNPP14] are again our inspiration in the following procedure which finds in β^9 computations the β^6 expected pairs of full state values that suits the truncated differentials path from Pattern 4 to 10 in Figure 4.

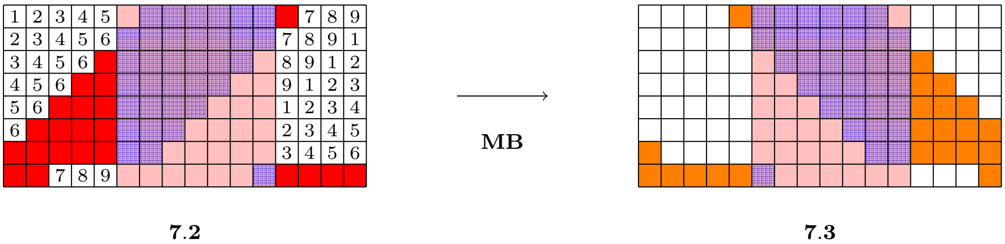


Figure 13: Merging lists, second edition.

Here, let b_i be in B_i, we denote by $b_{i,j}$ its component in the j^{th} row. For instance, $b_{8,3}$ is the component of b_8 in byte position in the 13^{th} column and in the 3^{rd} row of Pattern 7.2. As a precomputation, we sort lists B_1, B_2, B_3, B_5, B_8 and B_9 according respectively to $(b_{1,2}, b_{1,3}, b_{1,4}, b_{1,5})$, $(b_{2,3}, b_{2,4}, b_{2,5}, b_{2,6})$, $(b_{3,7})$, $(b_{5,1})$, $(b_{8,3})$ and $(b_{9,4}, b_{9,8})$. This is done in $O(\beta^5)$ computations:

$$B_1 = \bigcup_{b_1 \in B_1} B'_1(b_{1,2}, b_{1,3}, b_{1,4}, b_{1,5})\,, \quad B_3 = \bigcup_{b_3 \in B_3} B'_3(b_{3,7})\,, \quad B_8 = \bigcup_{b_8 \in B_8} B'_8(b_{8,3})\,,$$
$$B_2 = \bigcup_{b_2 \in B_2} B'_2(b_{2,3}, b_{2,4}, b_{2,5}, b_{2,6})\,, \quad B_5 = \bigcup_{b_5 \in B_5} B'_5(b_{5,1})\,, \quad B_9 = \bigcup_{b_9 \in B_9} B'_9(b_{9,4}, b_{9,8})\,.$$

By Assumption (2), $|B'_3| = \beta^3$, $|B'_5| = |B'_8| = |B'_9| = \beta^2$ and $|B'_1| = |B'_2| = 1$. By Gaussian elimination, constraints induced by Columns 3, 15 and 16 of Pattern 7.2 (red cells) and Pattern 7.3 (orange cells) can respectively be written as affine equations $l_3(b_3, b_4) =$

$r_3(b_5, b_6, b_7)$, $l_{15}(b_3, b_4) = r_{15}(b_5, b_8, b_9)$ and $l_{16}(b_3, b_4) = r_{16}(b_5, b_6, b_9)$. where l_i and r_i are affine expressions.

For each b_7 in B_7 (β elements), we compute the values of $b_{8,3}$, $b_{9,4}$, $b_{1,5}$, $b_{2,6}$ and $b_{3,7}$ induced by the constraints of Column 13. We compute and sort the list $B'_3(b_{3,7}) \times B_4$ of β^7 elements according to $(b_{3,6},\, b_{4,1},\, b_{4,7},\, l_3(b_3, b_4),\, l_{15}(b_3, b_4),\, l_{16}(b_3, b_4))$. By Assumption (2), B'_{34} have β elements:

$$B'_3(b_{3,7}) \times B_4 = \bigcup_{(b_3,b_4)\in B'_3(b_{3,7})\times B_4} B'_{34}(b_{3,6},\, b_{4,1},\, b_{4,7},\, l_3(b_3, b_4),\, l_{15}(b_3, b_4),\, l_{16}(b_3, b_4)).$$

For each b_8 in $B'_8(b_{8,3})$ (β^2 elements), for each b_6 in B_6 (β elements), we compute from (b_6, b_7, b_8) the values of $b_{5,1}$ and $b_{9,8}$ induced by constraints of Column 5.

For each b_5 in $B'_5(b_{5,1})$ (β^2 elements), for each b_9 in $B'_9(b_{9,4}, b_{9,8})$ (β^2 elements), we compute the values of $b_{3,6}$ and $b_{4,7}$ induced by constraints of Column 14 and the value of $b_{4,1}$ induced by constraint of Column 4.

For each (b_3, b_4) in $B'_{34}(b_{3,6},\, b_{4,1},\, b_{4,7},\, l_3(b_3, b_4),\, l_{15}(b_3, b_4),\, l_{16}(b_3, b_4))$ (β elements), we compute the values of $(b_{1,2},\, b_{1,3},\, b_{1,4},\, b_{1,5})$ and $(b_{2,3},\, b_{2,4},\, b_{2,5},\, b_{2,6})$ induced by constraints of Columns 13 to 16. We find the unique expected value of b_1 in $B'_1(b_{1,2},\, b_{1,3},\, b_{1,4},\, b_{1,5})$ and b_2 in $B'_2(b_{2,3},\, b_{2,4},\, b_{2,5},\, b_{2,6})$.

We store in a final list $\mathcal{L}$ all 7-uplet satisfying simultaneously the 3 independent equations induced by constraints of Columns 1 and 2. By Assumption (2), this happens with probability β^{-3}. After our enumeration of β^9 such elements, $\mathcal{L}$ has then cardinal β^6.

Overall computational complexity is then $O(\beta^9)$, corresponding to the number of elements we enumerate. Memory complexity is $O(\beta^7)$, corresponding to the cardinal of the list $B'_3 \times B_4$.

5 Conclusion and Perspectives

In this article, we present a new cryptanalysis method for internal permutations of Grøstl-512. This rebound attack variant takes advantage of a macroscopic description with SuperSBoxes and SuperLinear transformations to build a highly structured truncated differential path whose inbound phase involves 6 rounds while, to the best of our knowledge, all previously known methods controlled a maximum of 5 rounds. We obtain a distinguisher on 11 rounds with a complexity of about 2^{72} computations, improving upon the previously best known distinguisher, which attacks 10 rounds with a complexity of about 2^{392} computations. Even if the security of Grøstl-512 is not threatened, this attack requires a surprisingly low computational complexity.

We may remark that our attack do not run out all initial degrees of freedom of the 11-round differential path that we consider, since we expect that 2^{192} input pairs fulfill it. It is then natural to consider the closely related following 12-round differential path,

$$104 \xrightarrow{R_1} 53 \xrightarrow{R_2} 34 \xrightarrow{R_3} 34 \xrightarrow{R_4} 34 \xrightarrow{R_5} \cdots \xrightarrow{R_8} 34 \xrightarrow{R_9} 34 \xrightarrow{R_{10}} 53 \xrightarrow{R_{11}} 104 \xrightarrow{R_{12}} 128\,.$$

A quick analysis shows that 2^{16} input pairs should fulfill this differential path. Looking in this direction, we have not been able to design an algorithm which finds one with a reasonable calculation complexity. But in light of this, it is best to consider that permutations of this type must have at least 13 rounds to be considered as secure.

More generally, it can be interesting to consider other AES-based permutations in regard of highly structured truncated differential paths. We might hope that some of the techniques that we used in this attack are useful to obtain competitive distinguishers for these permutations too. We left this as an open problem.

References

[AMP10] Elena Andreeva, Bart Mennink, and Bart Preneel. On the indifferentiability of the Grøstl hash function. In *Security and Cryptography for Networks*, volume 6280 of *Lecture Notes in Comput. Sci.*, pages 88–105. Springer, 2010.

[BDPVA08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *Advances in Cryptology—EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Comput. Sci.*, pages 181–197. Springer, 2008.

[BFL11] Charles Bouillaguet, Pierre-Alain Fouque, and Gaëtan Leurent. Security analysis of SIMD. In *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Comput. Sci.*, pages 351–368. Springer, 2011.

[Dob99] Hans Dobbertin. Almost perfect nonlinear power functions on GF(2^n): The Welch case. *IEEE Trans. Information Theory*, 45(4):1271–1275, 1999.

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael*. Information Security and Cryptography. Springer, 2002.

[DR06] Joan Daemen and Vincent Rijmen. Understanding two-round differentials in AES. In *Security and Cryptography for Networks*, volume 4116 of *Lecture Notes in Comput. Sci.*, pages 78–94. Springer, 2006.

[FSZ09] Pierre-Alain Fouque, Jacques Stern, and Sébastien Zimmer. Cryptanalysis of tweaked versions of SMASH and reparation. In *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Comput. Sci.*, pages 136–150. Springer, 2009.

[Gil14] Henri Gilbert. A simplified representation of AES. In *Advances in Cryptology—ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Comput. Sci.*, pages 200–222. Springer, 2014.

[GKM$^+$09] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl - a SHA-3 candidate. In *Symmetric Cryptography*, number 09031 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[GP10] Henri Gilbert and Thomas Peyrin. Super-sbox cryptanalysis: Improved attacks for AES-like permutations. In *Fast Software Encryption*, volume 6147 of *Lecture Notes in Comput. Sci.*, pages 365–383. Springer, 2010.

[Gur17] Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*, 2017. Version 7.0.

[IPS13] Mitsugu Iwamoto, Thomas Peyrin, and Yu Sasaki. Limited-birthday distinguishers for hash functions - Collisions beyond the birthday bound can be meaningful. In *Advances in Cryptology—ASIACRYPT 2013*, volume 8870 of *Lecture Notes in Comput. Sci.*, pages 504–523. Springer, 2013.

[JF11] Jérémy Jean and Pierre-Alain Fouque. Practical near-collisions and collisions on round-reduced ECHO-256 compression function. In *Fast Software Encryption*, volume 6733 of *Lecture Notes in Comput. Sci.*, pages 107–127. Springer, 2011.

[JNPP13] Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Multiple limited-birthday distinguishers and applications. In *Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Comput. Sci.*, pages 533–550. Springer, 2013.

[JNPP14] Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Improved cryptanalysis of AES-like permutations. *J. Cryptology*, 27(4):772–798, 2014.

[JNPS12] Jérémy Jean, María Naya-Plasencia, and Martin Schläffer. Improved analysis of ECHO-256. In *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Comput. Sci.*, pages 19–36. Springer, 2012.

[Knu95] Lars R. Knudsen. Truncated and higher order differentials. In *Fast Software Encryption*, volume 1008 of *Lecture Notes in Comput. Sci.*, pages 196–211. Springer, 1995.

[LMR+09] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In *Advances in Cryptology—ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Comput. Sci.*, pages 126–143. Springer, 2009.

[MNPN+09] Krystian Matusiewicz, María Naya-Plasencia, Ivica Nikolić, Yu Sasaki, and Martin Schläffer. Rebound attack on the full Lane compression function. In *Advances in Cryptology—ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Comput. Sci.*, pages 106–125. Springer, 2009.

[MPRS09] Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer. Improved cryptanalysis of the reduced Grøstl compression function, ECHO permutation and AES block cipher. In *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Comput. Sci.*, pages 16–35. Springer, 2009.

[MRST09] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *Fast Software Encryption*, volume 5665 of *Lecture Notes in Comput. Sci.*, pages 260–276. Springer, 2009.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Information Security and Cryptology - 7th International Conference, INSCRYPT 2011*, volume 7537 of *Lecture Notes in Comput. Sci.*, pages 57–76. Springer, 2011.

[NP11] María Naya-Plasencia. How to improve rebound attacks. In *Advances in Cryptology—CRYPTO 2011*, volume 6841 of *Lecture Notes in Comput. Sci.*, pages 188–205. Springer, 2011.

[Pey07] Thomas Peyrin. Cryptanalysis of GRINDAHL. In *Advances in Cryptology—ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Comput. Sci.*, pages 551–567. Springer, 2007.

[SLW+10] Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. Non-full-active super-sbox analysis: Applications to ECHO and Grøstl. In *Advances in Cryptology—ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Comput. Sci.*, pages 38–55. Springer, 2010.

[WFWS10] Shuang Wu, Dengguo Feng, Wenling Wu, and Bozhan Su. Hyper-sbox view of AES-like permutations: A generalized distinguisher. In *Information Security and Cryptology—INSCRYPT 2010*, volume 6584 of *Lecture Notes in Comput. Sci.*, pages 155–168. Springer, 2010.

A A 11-Round Truncated Differential Path over Q_{1024}

Our MILP approach yields a similar highly structured truncated differential path for Q_{1024} which shares with P_{1024} the same macroscopic description. It is not very surprising as **Sh** transformation for Q_{1024}, only change in comparison with P_{1024}, is almost the same. The sequence of numbers of active bytes is the same as for P_{1024}. Figure 14 specifies in detail this truncated differential path, at a SuperSBox level.

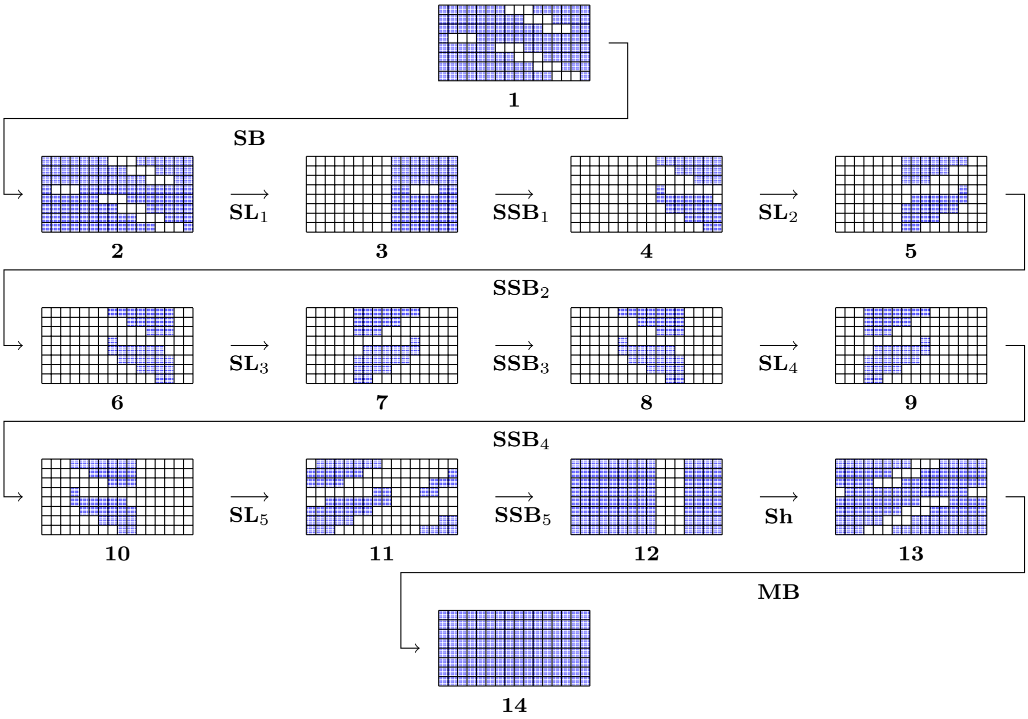


Figure 14: The 11−round truncated differential path over Q_{1024} with SuperSBoxes.