Efficient MILP Modelings for Sboxes and Linear Layers of SPN ciphers

Christina Boura, Daniel Coggia

IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. 3, pp. 327–361. DOI:10.13154/tosc.v2020.i3.327-361

Efficient MILP Modelings for Sboxes and Linear Layers of SPN ciphers

Christina Boura[1,2] and Daniel Coggia[1,3]

[1] Inria, Paris, France
Daniel.Coggia@inria.fr

[2] University of Versailles Saint-Quentin-en-Yvelines (UVSQ), Versailles, France
Christina.Boura@uvsq.fr

[3] Direction Générale de l'Armement, Paris, France

Abstract. Mixed Integer Linear Programming (MILP) solvers are regularly used by designers for providing security arguments and by cryptanalysts for searching for new distinguishers. For both applications, bitwise models are more refined and permit to analyze properties of primitives more accurately than word-oriented models. Yet, they are much heavier than these last ones. In this work, we first propose many new algorithms for efficiently modeling any subset of $\mathbb{F}_2^n$ with MILP inequalities. This permits, among others, to model differential or linear propagation through Sboxes. We manage notably to represent the differential behaviour of the AES Sbox with three times less inequalities than before. Then, we present two new algorithms inspired from coding theory to model complex linear layers without dummy variables. This permits us to represent many diffusion matrices, notably the ones of Skinny-128 and AES in a much more compact way. To demonstrate the impact of our new models on the solving time we ran experiments for both Skinny-128 and AES. Finally, our new models allowed us to computationally prove that there are no impossible differentials for 5-round AES and 13-round Skinny-128 with exactly one input and one output active byte, even if the details of both the Sbox and the linear layer are taken into account.

Keywords: MILP · Sbox · Linear Layer · Impossible Differential

1 Introduction

In symmetric-key cryptography, a popular technique for proving resistance against classical attacks is to model the behaviour of the cipher as a Mixed Integer Linear Programming (MILP) problem and solve it by some MILP solver. This method was applied for the first time by Mouha et al. [MWGP11] and by Wu and Wang [WW11] for finding the minimum number of differentially and linearly active Sboxes and provides in such a way a proof of resistance against these two classical attacks. Since then, the use of MILP not only by designers but also by cryptanalysts has increased, the advantage being that it is relatively easy to translate the cryptanalytic problem into linear constraints and use the available solvers to solve it.

For Substitution Permutation Networks (SPN), it is possible to get easy and relatively small models when searching for properties at the word level. For example, MILP is often used to search for a lower bound on the number of active Sboxes by exploring truncated differentials. However, wordwise models do not apply to all kind of ciphers (e.g. Present [BKL+07], Gift [BPP+17]) and most importantly they are much less accurate than models at bit level. For example, modeling the propagation of the differences

Received: 2020-06-01 Accepted: 2020-08-01 Published: 2020-09-28

bitwise and looking inside the Sboxes could yield longer impossible differentials than those discovered when only truncated differences are analyzed [ST17b].

Sun et al. [SHW+14a, SHW+14b] were the first to propose bit-oriented modelings for SPN ciphers. A non-trivial problem in doing so is to find an efficient representation of the valid differential propagations through an Sbox. Indeed, Sboxes are non-linear Boolean vectorial functions, therefore modeling their differential properties with $\mathbb{R}$-linear inequalities is not natural. Several approaches have been suggested to solve this problem. In the original works of Sun et al. [SHW+14b, SHW+14a] two different methods were notably proposed for modeling an Sbox. The first of them is a geometrical approach that consists in representing all possible transitions $(a \xrightarrow{S} b)$ through an n-bit Sbox as points $(a, b) \in \mathbb{R}^{2n}$ and then computing the H-representation of the convex hull of this set, that is all the geometric faces of the smallest convex containing it. The second method, based on a logical condition approach, consisted in representing by linear inequalities some conditional differential properties of the Sbox. Unfortunately, the problem of these two methods is that they are not efficient for large (e.g. 8-bit) Sboxes.

To solve this problem for large Sboxes, Abdelkhalek et al. [AST+17] observed that generating a minimal number of constraints in logical condition modeling can be converted into the problem of minimizing the product-of-sum representation of Boolean functions. This last problem is well-studied and algorithms for solving it exist, for example the *Quine-McCluskey* (QM) [Qui52, Qui55, McC56] or the *Espresso* [BSVMMH84] algorithms. In this way, Abdelkhalek et al. managed for the first time to generate linear constraints for 8-bit Sboxes, notably for the Sboxes of AES [aes01] and Skinny-128 [BJK+16]. While the number of linear constraints for the Sbox of Skinny provided in [AST+17] is as low as 372, the same method yields 8302 linear inequalities for the Sbox of AES, a modeling that is often too heavy to be used in practice.

Efficiently representing the Sboxes is a crucial part of the modeling process. But, a bad modeling of the diffusion layer can render the optimization process very slow or even impractical. Indeed, with the exception of some ciphers, e.g. Present [BKL+07], where the linear layer is just a bit-permutation, the diffusion is usually ensured by `XOR` gates. Yet, the `XOR` operation, while linear in $\mathbb{F}_2$ models very badly in $\mathbb{R}$. So, bitwise modeling of heavy linear layers, that need many `XOR`s to be represented, can lead to impractical systems with many linear inequalities or with many dummy variables.

The use of dummy variables in MILP models is a popular approach that needs to be discussed. To the best of our knowledge, there are three ways of using dummy variables: linear layer branch number modeling in wordwise modelings as introduced in [MWGP11], modeling $x_1 \oplus \cdots \oplus x_n = 0$ with a dummy integer variable t as $x_1 + \cdots + x_n = 2 \cdot t$ and finally using a dummy binary variable a for recording an intermediate state in the computation, for example modeling the above computation as $x_1 \oplus \cdots \oplus x_k \oplus a = 0$ and $a \oplus x_{k+1} \oplus \cdots \oplus x_n = 0$. For Sbox modelings, when the construction of the Sbox is known and well suited, only the latter can be helpful. However in general, in order to minimize the running time of the MILP solver, it is important to minimize the number of integer variables. In this paper we study bitwise modelings which need by definition many MILP integer variables. We hence restricted ourselves to not introducing dummy variables and leave the use of dummy integer variables for improving performance as an open problem.

Our Contributions In this work we propose several new bitwise MILP modelings for the propagation of differential properties through both Sboxes and linear layers. Our methods permit to efficiently model exact differential propagation through SPN ciphers and can be applied to prove resistance against differential cryptanalysis and its variants or to search for new differential-type attacks. Besides, the methods used for modeling the DDT of an Sbox are general enough for modeling an LAT or any Boolean function.

Modelings for large Boolean functions and Sboxes We introduce many different model-

ings for Boolean functions, by using algebraic or geometrical methods. With our techniques we manage to decrease significantly the number of inequalities needed to model large Sboxes. While Sasaki and Todo showed in [ST17a] that reaching the minimal number of inequalities is not necessarily the best approach for decreasing the solving time, yet a big difference in the number of inequalities leads to a real difference for the optimization process, as we demonstrate with experiments. We managed notably to represent the AES Sbox with 2882 inequalities, dividing by three the number of inequalities needed in the best previous approach.

Modeling matrices with entries in $\mathbb{F}_2$ We provide efficient modelings for linear layers without the use of dummy variables. We first explain why modeling the `XOR` of several binary variables, which is the central operation in most of the matrix/vector products over $\mathbb{F}_2$, needs many inequalities. Then, we introduce new algorithms inspired from coding theory that change the modeled matrix to significantly decrease this number.

Applications to impossible differential cryptanalysis from the designer's point of view We complete the work of Sasaki and Todo in [ST17b] and we use our modeling techniques for proving partial resistance against impossible differential cryptanalysis for AES and Skinny. More precisely, we show for both ciphers that even when the Sbox details are taken into account, there are no impossible differentials with one active input/output byte for 5-round AES and 13-round Skinny.

All of our MILP experiments were done with the `Gurobi` optimizer [GO20].

Organization The rest of the paper is organised as follows. Section 2 is dedicated to our different methods for modeling Boolean functions and Sboxes. Then, in Section 3 we present an algorithm to efficiently model linear layers. In Section 4, we run an experiment for illustrating the effectiveness of the previously-introduced methods. Finally, we apply our new models to impossible differential cryptanalysis in Section 5.

2 MILP Modeling for Boolean functions and Sboxes

The methods to be introduced in this section can be applied to characterize any set $P \subset \{0,1\}^m$ with $\mathbb{R}$-linear inequalities. As any such subset $P \subset \{0,1\}^m$ can be seen as the support of some Boolean function $f_P : \mathbb{F}_2^m \to \mathbb{F}_2$ operating on m bits, the inequalities representing P model the constraint $f_P(x_0, \ldots, x_{m-1}) = 1$. However, as our main target is the modeling of differential, linear or other behaviours through an Sbox $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$, and such behaviours can be represented (as will be explained below) by some Boolean function, we will describe some of our techniques, without loss of generality, through the spectrum of the differential behaviour of an Sbox.

In what follows, a (differential) transition $x \to y$ through the Sbox will be seen as a vector of $\mathbb{F}_2^m$, involving $m = 2n$ binary variables and will be represented, depending on the context, either as $(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1})$ or as $(x_0, \ldots, x_{m-1})$. The Hamming weight of a vector $x \in \mathbb{F}_2^m$, i.e. the number of its non-zero bits, will be denoted by $\mathrm{wt}(x)$. Finally, we will often use the notation $[a, b]$ to represent the set of all integers k such that $a \leq k \leq b$.

2.1 Modeling Boolean functions and DDTs

For many problems, as for example the search for good differential characteristics, bitwise modelings are more often adapted than wordwise ones, as they are more precise and permit to follow information propagation at the bit level. In such models, a binary variable

is assigned to each bit of a differential characteristic. A variable has the value 1 if the corresponding bit has a difference in the characteristic — in which case it is called active — and 0 otherwise. In a MILP model, in order to follow how the information is propagated through the different components of the cipher, each different layer has to be efficiently modeled. For Sboxes, this is typically done by looking at the Difference Distribution Table (DDT), that is a $2^n \times 2^n$ table given by

$$\mathrm{DDT}(a,b) = \#\left\{ x \in \mathbb{F}_2^n \,\middle|\, S(x) \oplus S(x \oplus a) = b \right\}.$$

When one only cares whether a differential propagation is possible or not, it is enough to model the Boolean function

$$\begin{array}{ccc} \mathbb{F}_2^{2n} & \to & \mathbb{F}_2 \\ (x,y) & \mapsto & \begin{cases} 0, & \text{if } \mathrm{DDT}(x,y) = 0, \\ 1, & \text{otherwise.} \end{cases} \end{array}$$

However, if one wants to take into account the different integer values of the DDT, the authors of [AST+17] propose to encode this information by modeling instead the Boolean functions

$$\begin{array}{ccc} \mathbb{F}_2^{2n} & \to & \mathbb{F}_2 \\ (x,y) & \mapsto & \begin{cases} 0, & \text{if } \mathrm{DDT}(x,y) \neq p, \\ 1, & \text{otherwise,} \end{cases} \end{array}$$

for each value $p > 0$ such that $\exists (a,b) : \mathrm{DDT}(a,b) = p$.

As one can see, modeling a DDT is equivalent to modeling some specific Boolean function. From now on, we will focus on modeling general Boolean functions but all our examples and applications will be focused on DDTs. In all these examples we will be only interested in whether differential propagations are possible or not, without caring about their concrete probability. However, one has to remember, that all these methods can be applied to any other Boolean function and in particular to other cryptanalysis techniques whose properties can be described through some table, as are the linear and the boomerang cryptanalysis [Wag99].

Lets take the example of the following 3-bit Sbox, used inside the block cipher PrintCipher [KLPR10]:

x	0	1	2	3	4	5	6	7
S(x)	0	1	3	6	7	4	5	2

Its DDT can be visualized in Table 1.

Table 1: DDT of the 3-bit Sbox used in PrintCipher

	0	1	2	3	4	5	6	7
0	8	0	0	0	0	0	0	0
1	0	2	0	2	0	2	0	2
2	0	0	2	2	0	0	2	2
3	0	2	2	0	0	2	2	0
4	0	0	0	0	2	2	2	2
5	0	2	0	2	2	0	2	0
6	0	0	2	2	2	2	0	0
7	0	2	2	0	2	0	0	2

Then, by denoting by x_0, x_1, x_2 the three bits of the input difference x and by y_0, y_1, y_2 the three bits of the output difference y, where x_0 and y_0 are the LSBs of x and y respectively, the following system of 7 inequalities is satisfied by all the valid transitions while removing the 35 impossible transitions $x \to y$ for which $\mathrm{DDT}(x,y) = 0$.

$$\begin{array}{ll} -2x_0 - 2x_1 + x_2 - 2y_0 - 2y_1 + y_2 \geq -6 & x_0 + 2x_1 + 4x_2 + 3y_0 + 2y_1 - 4y_2 \geq 0 \\ -2x_0 + x_1 - 2x_2 - 2y_0 + y_1 - 2y_2 \geq -6 & -3x_0 + 2x_1 - x_2 + 4y_0 + 2y_1 + 4y_2 \geq 0 \\ x_0 - 2x_1 - 2x_2 + y_0 - 2y_1 - 2y_2 \geq -6 & 4x_0 - 2x_1 + x_2 - 2y_0 + 4y_1 + 3y_2 \geq 0 \end{array} \quad (1)$$

2.2 State of the art

Given the truth table of a Boolean function f, the question is how to efficiently model the constraint $f(x) = 1$ by a system of $\mathbb{R}$-linear inequalities. This problem can then be divided into two sub-problems:

Problem 1 How to generate a (possibly large) set of inequalities on variables $x_0, \ldots, x_{m-1}$ that correctly models f?

Problem 2 How to choose a (typically much smaller) subset of this set of inequalities that still correctly represents f but leads to more efficient MILP models?

For differential cryptanalysis, the above general problem corresponds to modeling the fact that $(x_0, \ldots, x_{n-1}) \to (x_n, \ldots, x_{2n-1})$ is a possible transition in a DDT. To solve *Problem 1*, two different approaches were proposed in 2014 by Sun et al. [SHW+14a, SHW+14b]. The first is a geometrical one and consists in computing the H-representation of the convex hull of the set of possible transitions. The second one is based on logical condition modeling. Below, we briefly explain these two methods.

The method of the H-representation of the convex hull consists, as its name suggests, in computing the H-representation of the convex hull of all possible points $a \in \mathbb{F}_2^m$ such that $f(a) = 1$ seen as vectors of $\mathbb{R}^m$. Taking then the $(m-1)$-dimensional faces of the convex hull yields a correct set of inequalities. The H-representation can be for example computed through an algebra computer system such as `Sage` [The20] and gives a system of linear inequalities excluding all impossible points (e.g. all points a such that $f(a) = 0$).

The second method proposed by Sun et al. and called logical condition modeling is based on the idea that each impossible point can be removed by a single inequality in a simple way. Indeed, consider an impossible point $a = (a_0, \ldots, a_{m-1})$. Then, the inequality

$$\sum_{i=0}^{m-1} (1 - a_i)x_i + a_i(1 - x_i) \geq 1 \quad (2)$$

only discards this point a. Lets take as example the DDT of PrintCipher. As it can be seen from Table 1, `(0x1,0x6)` is an impossible transition through the DDT. By writing the input and output in a bitwise manner we have that $(100) \nrightarrow (011)$. The above formula gives the inequality $-x_0 + x_1 + x_2 + y_0 - y_1 - y_2 \geq -2$ that is satisfied by all points in $\mathbb{F}_2^6$ but `(0x1,0x6)`. This method can then be applied to all impossible transitions $x \to y$ and yields easily a system of inequalities containing as many constraints as the number of impossible transitions through the DDT, or as the number of zeros of the Boolean function in the general case.

However, as mentioned in [AST+17], both these methods that provide a solution for *Problem 1* have the disadvantage of not being efficient for modeling 8-bit Sboxes. For the first one, computing the H-representation of the convex hull for such big Sboxes is nearly impossible, while the second method yields a very high number of initial inequalities with by construction no hope for a correct subset for *Problem 2*. For example, the Skinny-128 Sbox has 54067 impossible transitions and this number of corresponding inequalities is too high to represent the Sbox for any related MILP problem.

In 2017, Abdelkhalek et al. [AST+17] made an important step forwards in the logical condition modeling direction, by translating the problem of searching for good inequalities

for 8-bit Sboxes into the classical problem of minimization of the product-of-sum representation of the related Boolean function and by using the Quine-McCluskey (QM) algorithm to solve it. This method permits to solve at once the two steps of the Sbox modeling: Find many good inequalities (the prime implicants in the QM vocabulary) and keep among them a good representative set. In the case of QM this representative set corresponds to the minimal number of equations. The problem however of QM is that in practice it needs high memory ressources and it can be slow. For this reason, Abdelkhalek et al. used another algorithm, called the *Espresso* algorithm, a heuristic method for minimizing the number of terms in a product-of-sum representation. Espresso is not guaranteed to find the minimum, and it usually doesn't in the case of 8-bit Sboxes, but its solutions are good enough to be used in practice.

No matter the method used for solving *Problem 1*, one must choose among the initial set of inequalities a good representative set for representing the support of the Boolean function. This is what we called as *Problem 2*. As mentioned in [ST17a], determining how many and which inequalities to keep is not an evident decision. This step is however necessary, as in both methods from [SHW+14a, SHW+14b] and all the new methods that we are going to present, the number of generated inequalities is high and has an important impact on the optimization time. For example, in the convex-hull method, the number of linear inequalities that `Sage` returns is typically quite high, containing notably many redundant ones. The authors of [SHW+14b] applied then a greedy algorithm for solving *Problem 2*. At each step, this algorithm adds to the solution set the best possible inequality, that is the inequality removing the highest number of points among those that have not been removed yet. A nice approach for solving this step was later given by Sasaki and Todo in [ST17a]. They proposed to model the problem of minimizing the set of inequalities that remove all the impossible propagation points as a MILP problem itself and solve it by some solver. More precisely, their method consists in assigning a binary variable z_i to each inequality found by solving *Problem 1*. Then for each impossible point p, it is required that at least one inequality removing p is chosen with $\sum_{i \text{ s.t. ineq. } i \text{ removes } p} z_i \geq 1$. Finally the MILP solver is used for minimizing $\sum_i z_i$ and $\left\{ i \,\middle|\, z_i = 1 \right\}$ gives a solution of *Problem 2*.

It appears however in [ST17a] that the smallest subset of inequalities found by this approach that correctly models a DDT, will not necessarily provide the overall best performance when running a complete cipher modeling. Moreover, this auxiliary MILP problem can be too heavy when the initial set of inequalities is large. In our experiments, we have found the greedy approach from [SHW+14b] to provide better subsets for performance even if they are a bit larger. We hence used it in the applications. However, we consider the minimization approach from [ST17a] in solving *Problem 2* to be a good benchmarking indicator for methods solving *Problem 1*.

In the remaining part of the section we analyze in depth the problem of efficiently modeling a Boolean function with MILP inequalities and we concentrate on *Problem 1*. Indeed our goal is to generate efficient algorithms for modeling a Boolean function by using the smallest number of inequalities as an indicator for the quality of the method.

We propose different methods for doing so. The first method, based on generating better inequalities from the H-representation is applicable for up to 12-bit Boolean functions and gives better results than all previous methods for up to 6-bit Sboxes. Unfortunately, this method is not applicable for 8-bit Sboxes. For this reason, we develop other methods for larger Boolean functions and Sboxes. With these methods, described in Sections 2.4 and 2.5 we manage to model 8-bit Sboxes with a much smaller number of inequalities than what was done before.

In the remaining part of this section, and to facilitate comprehension, all methods will be described through the DDT modeling application. Nevertheless, none of these techniques is DDT-specific and they can be applied directly for modeling any Boolean function.

2.3 Convex hull techniques for up to 6-bit Sboxes

When the computation of the H-representation of the convex hull of all possible transitions in a DDT is computationally feasible, this H-representation provides by definition a set of inequalities modeling the possible differential transitions through the Sbox. However, as we will show, it is possible to compute many other linear inequalities from this initial set by simply adding up some of them. The reason for doing so is to generate potentially better inequalities than the ones directly given by the convex hull, where better means that the new inequalities remove more impossible transitions than the initial ones do.

Indeed, if a possible differential transition $z = (x, y) \in \{0, 1\}^m$, with $m = 2n$ satisfies the k inequalities $C_1, \ldots, C_k : c_0^\ell z_0 + \cdots + c_{m-1}^\ell z_{m-1} + b_\ell \geq 0$, with $\ell \in [1, k]$ then it obviously also satisfies the inequality

$$\big(\sum_{i=1}^{k} c_0^i\big) z_0 + \cdots + \big(\sum_{i=1}^{k} c_{m-1}^i\big) z_{m-1} + \sum_{i=1}^{k} b_i \geq 0$$

produced by simply summing up the initial k inequalities and denoted in the sequel as $C_{new} = C_1 + \ldots + C_k$.

Of course, most of the inequalities produced by randomly summing k inequalities from the H-representation of the convex hull, do not present any interest, as they will very probably be satisfied by the whole space $\{0, 1\}^m$. In order to produce meaningful new linear inequalities from the H-representation of the convex hull, we noticed that if k hyperplanes of the H-representation share a vertex on the cube $\{0, 1\}^m$, (i.e. a possible transition), then the addition of the k corresponding inequalities will probably yield an interesting new constraint, given that its hyperplane intersects with the cube at least on this particular vertex. By "interesting" we mean here that the new inequality C_{new} will remove a different (potentially larger) set of impossible transitions than the original inequalities. This idea is illustrated in Figure 1 (Appendix A) and described by Algorithm 1.

Algorithm 1 takes as input a set $\mathcal{S}_{\text{pos}}$ corresponding to the possible transitions through an Sbox S and a parameter k that indicates the number of inequalities to be added together each time. Then it starts by generating the convex hull corresponding to $\mathcal{S}_{\text{pos}}$ by using for example the `inequality_generator()` function of the `sage.geometry.polyhedron` class of the `Sage` computer algebra system [The20]. This gives us an initial set of inequalities $\mathcal{C}_{\text{set}}$ of the form $c_0 x_0 + \cdots + c_{m-1} x_{m-1} + b_\ell \geq 0$. Then, for every point p in $\mathcal{S}_{\text{pos}}$ we add together any k inequalities $C_1, \ldots, C_k$ that this point satisfies with a zero left-hand side, i.e. p belongs to the hyperplanes defined by those inequalities. We add in $\mathcal{C}_{\text{set}}$ the new inequality only if it is meaningful, in the sense that it removes a new set of impossible transitions.

In practice, Algorithm 1 is rather fast for 4-bit Sboxes — a few minutes for $k = 2$ and a few hours at most for $k = 3$ — and the set of constraints $\mathcal{C}_{\text{set}}$ obtained that way does not have too many elements, which allows fast minimization in solving *Problem 2*. For example, `Sage` returns 327 linear inequalities for the Sbox of PRESENT. By applying Algorithm 1 with $k = 2$ we get a little bit less than 500 inequalities, and for $k = 3$ we get a little bit less than 700 inequalities.

We applied Algorithm 1 to solving *Problem 1* for different Sboxes from the literature and benchmarked it by solving *Problem 2* with the method of [ST17a] to obtain a minimal modeling. The results are summarized in Table 2. We took $k = 2$ to run Algorithm 1, apart for TWINE, PRIDE, SERPENT S3 and SERPENT S7 where taking $k = 3$ gave slightly better results. The case $k = 1$, corresponds to the method of [ST17a]. Even as already said, the exact minimum is not necessarily what one has to take to minimize the solving time of the global problem, it is however a good indicator of the quality of the method used to solve *Problem 1* and permits to compare the different methods between them. We nonetheless also give the results with the greedy approach from [SHW+14a] and $k = 1$ for

Algorithm 1 Compute a set of inequalities from possible transitions.

procedure ComputeConstraints($\mathcal{S}_{\text{pos}}$, k)
  $\mathcal{H}_{\text{set}} \leftarrow \text{Hull}(\mathcal{S}_{\text{pos}})$
  $\mathcal{C}_{\text{set}} \leftarrow \mathcal{H}_{\text{set}}$
  for all $p \in \mathcal{S}_{\text{pos}}$ **do**
    for all $\{C_1, \ldots, C_k\} \in \mathcal{P}(\mathcal{H}_{set})$ such that p belongs to the hyperplanes of $C_1, \ldots, C_k$ **do**
      $C_{\text{new}} = C_1 + \ldots + C_k$
      if C_{new} removes a new set of impossible transitions **then**
        $\mathcal{C}_{\text{set}} \leftarrow \mathcal{C}_{\text{set}} \cup \{C_{\text{new}}\}$
      end if
    end for
  end for
end procedure

completeness. As can be seen from Table 2 but also for all the Sboxes we tested for $n \leq 6$, running the algorithm with $k > 1$ always gave better results than with $k = 1$.

Unfortunately, for larger Sboxes, and notably for $n = 8$ computing the convex hull is computationally hard. For this reason, we describe in the following sections, alternative methods that can be used for modeling 8-bit Sboxes.

Table 2: Number of inequalities to model differential transitions for various 4-bit Sboxes

	# Inequalities				# Inequalities		
Sbox	[SHW+14a]	[ST17a]	Alg. 1	Sbox	[SHW+14a]	[ST17a]	Alg. 1
Present	22	21	**17**	Serpent S0	23	21	**17**
Klein	22	21	**19**	Serpent S1	24	21	**17**
Twine	23	23	**19**	Serpent S2	25	21	**18**
Prince	26	22	**19**	Serpent S3	31	27	**20**
Piccolo	23	21	**16**	Serpent S4	26	23	**19**
MIBS	27	23	**20**	Serpent S5	25	23	**19**
LBlock S0	28	24	**17**	Serpent S6	22	21	**17**
LBlock S1	27	24	**17**	Serpent S7	30	27	**20**
LBlock S2	27	24	**17**	Lilliput	–	23	**19**
LBlock S3	27	24	**17**	Minalpher	–	22	**19**
LBlock S4	28	24	**17**	Midori S0	–	21	**16**
LBlock S5	27	24	**17**	Midori S1	–	22	**20**
LBlock S6	27	24	**17**	Rectangle	–	21	**17**
LBlock S7	27	24	**17**	Skinny	–	21	**16**
LBlock S8	28	24	**17**	Gift	–	–	**17**
LBlock S9	27	24	**17**	Pride	–	–	**16**

2.4 Logical condition techniques for 8-bit Sboxes

In this first section, we show that one can easily derive simple inequalities to remove spaces of the form $a \oplus \text{Prec}(u)$ inside the DDT. Let again $m = 2n$ where n is the bit-size of the Sbox. For some $u = (u_0, u_1, \ldots, u_{m-1}) \in \mathbb{F}_2^m$ we denote by $\text{supp}(u) = \left\{ i \,\middle|\, u_i = 1 \right\} \subseteq [0, m-1]$. Furthermore $\text{Prec}(u)$ denotes the space $\left\{ x \in \mathbb{F}_2^m \,\middle|\, x \preceq u \right\}$, where $x \preceq u$ means that $x_i \leq u_i$ for all $i \in [0, m-1]$.

Proposition 1. *Let $a \in \mathbb{F}_2^m$ and $u \in \mathbb{F}_2^m$ such that $\text{supp}(a) \bigcap \text{supp}(u) = \emptyset$ and let $I = [0, m-1] \setminus \big(\text{supp}(a) \bigcup \text{supp}(u)\big)$. Then, for all $x \in \mathbb{F}_2^m$,*

$$-\sum_{i \in \text{supp}(a)} x_i + \sum_{i \in I} x_i \geq 1 - \text{wt}(a) \Leftrightarrow x \notin a \oplus \text{Prec}(u).$$

Proof. Let $x \in \mathbb{F}_2^m$. If $x \in a \oplus \text{Prec}(u)$,

$$-\sum_{i \in \text{supp}(a)} x_i + \sum_{i \in I} x_i = -\sum_{i \in \text{supp}(a)} a_i = -\text{wt}(a).$$

Otherwise $x \oplus a \not\preceq u$, so there exists some $\ell \in \text{supp}(a) \bigcup I$ such that $x_\ell = 1 - a_\ell$.

- If $\ell \in \text{supp}(a), x_\ell = 0$ and $\sum_{i \in \text{supp}(a)} x_i \leq \text{wt}(a) - 1$.
- If $\ell \in I, x_\ell = 1$ and $\sum_{i \in I} x_i \geq 1$.

In both cases, $-\sum_{i \in \text{supp}(a)} x_i + \sum_{i \in I} x_i \geq 1 - \text{wt}(a)$. □

Example 1. We show how the above method can be applied to remove some invalid transitions for the Sbox of the block cipher PRESENT [BKL+07]. The Sbox used in PRESENT is a 4-bit permutation $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ given by the following table:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	12	5	6	11	9	0	10	13	3	14	15	8	4	7	1	2

The Difference Distribution Table (DDT) of PRESENT is given in Appendix B.

Here $m = 2 \times 4 = 8$. For better visualizing points in the DDT, we will see $\mathbb{F}_2^8$ as $\mathbb{F}_2^4 \times \mathbb{F}_2^4$. Further, we will use the following bit ordering. For a point $[\alpha, \beta] \in \mathbb{F}_2^4 \times \mathbb{F}_2^4$, the index 0 will correspond to the least significant bit (LSB) of α, 3 will correspond to the most significant bit (MSB) of α, 4 to the LSB of β and 7 to the MSB of β. Let $a = \texttt{[0,1]}, u = \texttt{[9,4]} \in \mathbb{F}_2^4 \times \mathbb{F}_2^4$. Then,

$$\text{Prec}(u) = \{[0,0], [0,4], [1,0], [1,4], [8,0], [8,4], [9,0], [9,4]\}.$$

Further, as $\text{supp}(a) = \{4\}$ and $\text{supp}(u) = \{0, 3, 6\}$, $I = \{1, 2, 5, 7\}$. Therefore the equation $-x_4 + x_1 + x_2 + x_5 + x_7 \geq 0$ removes exactly the 8 points in the space

$$a \oplus \text{Prec}(u) = \{[0,1], [0,5], [1,1], [1,5], [8,1], [8,5], [9,1], [9,5]\}.$$

We can verify from the DDT in Appendix B that all these points correspond indeed to invalid transitions through the DDT.

2.4.1 An efficient algorithm for searching for spaces of the form $a \oplus \text{Prec}(u)$

Given a set of impossible transitions $\mathcal{P}$, Algorithm 2 finds all subsets of the form $a \oplus \text{Prec}(u)$ excluding those that are subsets of others. For each $a \in \mathcal{P}$, it builds spaces $a \oplus \text{Prec}(u) \subseteq \mathcal{P}$ by progressively incrementing the weight of u, with u such that $a \oplus u \in \mathcal{P}$ and $\text{supp}(u) \cap \text{supp}(a) = \emptyset$ and checking for all $v \preceq u, \text{wt}(v) = \text{wt}(u) - 1$ whether $a \oplus \text{Prec}(v)$ has already been identified as a subset of $\mathcal{P}$.

We show next that Algorithm 2 and the Quine-McCluskey algorithm are strongly related. The Quine-McCluskey algorithm has two steps. Given a set of points $\mathcal{P}$, the first step finds a set $\mathcal{S}$ of subspaces $a \oplus \text{Prec}(u) \subseteq \{0,1\}^m$ such that

$$\mathcal{P} = \bigcup_{a \oplus \text{Prec}(u) \in \mathcal{S}} a \oplus \text{Prec}(u),$$
$$\forall (a \oplus \text{Prec}(u)) \subseteq \mathcal{P}, \exists E \in \mathcal{S} : a \oplus \text{Prec}(u) \subseteq E,$$
$$\text{and } \forall E, F \in \mathcal{S}, E \not\subseteq F.$$

Algorithm 2 Find $a \oplus \mathrm{Prec}(u)$ sets included in the set $\mathcal{P} \subset \{0,1\}^m$.

procedure AffinePrec($\mathcal{P}$)
  $\mathcal{S}_{\text{out}} \leftarrow \emptyset$ ▷ Output set
  for all $a \in \mathcal{P}$ **do**
    $\mathcal{S}_{\text{interesting}} \leftarrow \emptyset$ ▷ Set of interesting new inequalities of the form $a \oplus \mathrm{Prec}(u)$
    for all $i \in [0, m]$ **do**
      $\mathcal{S}_i \leftarrow \emptyset$
      $\mathcal{U}_i \leftarrow \emptyset$
    end for
    ▷ We start by grouping all u such that $a \oplus \mathrm{Prec}(u) \subseteq \mathcal{P}$
    ▷ and $\mathrm{supp}(a) \cap \mathrm{supp}(u) = \emptyset$ by their weight in sets $\mathcal{U}$.
    for all $p \in \mathcal{P}$ **do**
      $u \leftarrow a \oplus p$
      if $\mathrm{supp}(a) \cap \mathrm{supp}(u) = \emptyset$ **then**
        $\mathcal{U}_{\mathrm{wt}(u)} \leftarrow \mathcal{U}_{\mathrm{wt}(u)} \cup \{u\}$
      end if
    end for
    ▷ If $\mathcal{U}_1 \neq \emptyset$, $a \oplus \mathrm{Prec}(0)$ is no longer interesting
    ▷ since $\forall u \in \mathcal{U}_1, a \in a \oplus \mathrm{Prec}(u)$.
    if $\mathcal{U}_1 = \emptyset$ **then**
      $\mathcal{S}_{\text{interesting}} \leftarrow \left\{ a \oplus \mathrm{Prec}(u) \,\middle|\, u \in \mathcal{U}_0 \right\}$
    else
      $\mathcal{S}_{\text{interesting}} \leftarrow \left\{ a \oplus \mathrm{Prec}(u) \,\middle|\, u \in \mathcal{U}_1 \right\}$
    end if
    ▷ $a \oplus \mathrm{Prec}(u) \in \mathcal{S}_k \Leftrightarrow \mathrm{wt}(u) = k$ and $a \oplus \mathrm{Prec}(u) \subseteq \mathcal{P}$
    $\mathcal{S}_0 \leftarrow \left\{ a \oplus \mathrm{Prec}(u) \,\middle|\, u \in \mathcal{U}_0 \right\}$
    $\mathcal{S}_1 \leftarrow \left\{ a \oplus \mathrm{Prec}(u) \,\middle|\, u \in \mathcal{U}_1 \right\}$
    for $k \in [2, m]$ **do**
      for $u \in \mathcal{U}_k$ **do**
        if $\forall v \preceq u$ st. $\mathrm{wt}(v) = k-1,\ a \oplus \mathrm{Prec}(v) \in \mathcal{S}_{k-1}$ **then**
          $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{a \oplus \mathrm{Prec}(u)\}$
          for all $v \preceq u$ st. $\mathrm{wt}(v) = k-1$ **do**
            ▷ Remove $a \oplus \mathrm{Prec}(v)$ since $a \oplus \mathrm{Prec}(v) \subset a \oplus \mathrm{Prec}(u)$
            $\mathcal{S}_{\text{interesting}} \leftarrow \mathcal{S}_{\text{interesting}} \backslash \{a \oplus \mathrm{Prec}(v)\}$
          end for
        end if
      end for
      $\mathcal{S}_{\text{interesting}} \leftarrow \mathcal{S}_{\text{interesting}} \cup \mathcal{S}_k$
    end for
    $\mathcal{S}_{\text{out}} \leftarrow \mathcal{S}_{\text{out}} \cup \mathcal{S}_{\text{interesting}}$
  end for
  return $\mathcal{S}_{\text{out}}$
end procedure

This is exactly what does Algorithm 2. The second step of QM then searches for a minimal set $\mathcal{S}' \subseteq \mathcal{S}$, in the sense that:

$$\mathcal{P} = \bigcup_{a \oplus \mathrm{Prec}(u) \in \mathcal{S}'} a \oplus \mathrm{Prec}(u) \quad \text{and} \quad \forall E, F, G \in \mathcal{S}', E \not\subseteq F \cup G.$$

This formulation of the Quine-McCluskey algorithm is very different from the one encountered in the literature and in the best of our knowledge, it is the first time that

this algorithm is presented in this way. We use this presentation as it is well suited for understanding the link between the two methods.

The important thing for us is that we principally need this first step to find good modelings. This corresponds to solving *Problem 1*. The second step of the QM algorithm, corresponds to providing a solution for *Problem 2*, by minimizing the number of terms with the objective of finding a good circuit for a Boolean function, but not necessarily a good MILP modeling. Moreover, this second step is computationally harder than the first one and acts as a bottleneck when using QM as a black box inequality generator for MILP modelings. Indeed, it is much faster to use Algorithm 2 alone for solving *Problem 1* together with a greedy algorithm or a MILP-based algorithm for solving *Problem 2*. This is not only faster but can also provide a significantly lower number of inequalities.

We notably applied Algorithm 2 for generating an initial set of inequalities for the 8-bit Sboxes of AES and Skinny-128. We then obtained 70336 initial inequalities for AES and 8829 for Skinny-128. The running time was of around 15 minutes for AES and 2 hours for Skinny-128 where 90% of the time was spent not for finding new inequalities but for removing not interesting ones (lines 26-28 in Algorithm 2). The difference in these running times is explained by the fact that as the DDT of Skinny-128 is very sparse, there are many more possible spaces $a \oplus \mathrm{Prec}(u)$ than for the DDT of AES. After this, to find a representative set among these initial inequalities (*Problem 2*) we set up a minimization problem and solved it with `Gurobi`. While for Skinny-128 the problem was solved in just a few seconds providing us with the global minimal (which is thus the same as the QM algorithm), for AES the problem didn't reach the minimum even after 1900975 seconds of search on a 8-core laptop. However, the solution provided by `Gurobi`, even if not the minimal one is much better than the one given by Espresso, as it can be seen in Table 3. Furthermore, according to the authors of [AST+17], QM itself cannot be applied to AES because of its memory complexity.

Table 3: Number of inequalities to exclude $\mathcal{P}$ for various 8-bit S-boxes with the methods of [AST+17] and Algorithm 2.

		# Inequalities		
		[AST+17]		Alg. 2 with
`Sbox`	# Zero entries	QM	Espresso	MILP minimization
AES	33150	–	8302	**7461**
Skinny-8	54067	**372**	376	**372**

While for Sboxes with sparse DDTs, as the one of Skinny-128, the method of this section provides a quite compact modeling, for Sboxes with low differential uniformity (i.e. highest value in the DDT), as the one of AES the number of obtained inequalities is quite high for practical applications. For this reason, we provide in the following sections new methods for modeling large Sboxes that outperform in most of the cases the methods provided up to now.

2.5 Modeling an Sbox with inequalities issued from balls $\mathcal{B}(d, c)$

We saw in the previous section that each set of impossible transitions of the form $a \oplus \mathrm{Prec}(u)$ provided a single inequality for removing all points in the set. What we learned in particular from this is that it is interesting to group impossible transitions in sets having a particular algebraic description and get inequalities out of them. In this section, we investigate a different type of sets and show how to use their mathematical description to get nice inequalities. More precisely, we show that points lying in a ball $\mathcal{B}(d, c)$ of radius d centred at a point $c \in \mathbb{F}_2^m$, can be removed together by a simple inequality.

Definition 1. A *ball* of $\mathbb{F}_2^m$ of radius d centred at $c \in \mathbb{F}_2^m$ is the subset of all points whose Hamming distance from the center c is at most d: $\mathcal{B}(d,c) = \left\{ x \in \mathbb{F}_2^m \,\middle|\, \mathrm{wt}(x \oplus c) \leq d \right\}$. Furthermore, by $\mathcal{S}(d,c)$ we will denote the *sphere* of radius d centred on c, that is the set of points $x \in \mathcal{B}(d,c)$ for which $wt(x \oplus c) = d$.

To illustrate this idea, we start with an example of the most simple case — balls of radius 1.

Example 2. Consider $m = 4$ and let $\mathcal{B}(1,c)$ be a ball of radius 1 centred at $c = (1,0,0,0) \in \mathbb{F}_2^4$: $\mathcal{B}(1,c) = \{(1,0,0,0),(0,0,0,0),(1,1,0,0),(1,0,1,0),(1,0,0,1)\}$.

It can be checked that all five points of the above ball can be removed by

$$(1-x_0) + x_1 + x_2 + x_3 \geq 2.$$

We can construct similar examples for any ball of dimension $d > 1$. As we will show now, for any dimension m and any point $c \in \mathbb{F}_2^m$, it is possible to construct an inequality that removes all points of the ball $\mathcal{B}(d,c)$.

Proposition 2. *Let $c \in \mathbb{F}_2^m$. Then, the inequality*

$$\sum_{i=0}^{m-1} (1-c_i)x_i + c_i(1-x_i) \geq d+1, \tag{3}$$

holds if and only if $x \notin \mathcal{B}(d,c)$.

Proof. Notice here that $\sum_{i=0}^{m-1}(1-c_i)x_i + c_i(1-x_i) = \sum_{i=0}^{m-1} x_i \oplus c_i = \mathrm{wt}(x \oplus c)$. For any point $u \in \mathcal{B}(d,c)$, we thus have $\sum_{i=0}^{m-1}(1-c_i)u_i + c_i(1-u_i) = \mathrm{wt}(u \oplus c) \leq d$. On the other side, Eq. (3) is satisfied for any point $u \in \mathbb{F}_2^m \setminus \mathcal{B}(d,c)$, as $\mathrm{wt}(u \oplus c) > d$.

□

2.6 Distorted balls

When searching for inequalities removing impossible transitions for a DDT, we have to be sure that the corresponding ball does not contain any possible transitions that we would mistakenly remove. In Sboxes used in practice, notably those with a low differentially uniformity, as the number of non-zero coefficients is usually large, removing entire balls does not usually work, as the number of balls for which all points correspond to impossible transitions is typically very small. While the above method works well for sparse DDTs as the one of Skinny-128, for the Sbox of Present, no plain ball, even of radius 1, can be removed with this method.

We will show now, that we can still extract an inequality from a ball for which we have removed from its edge all possible transition points. We call such a ball *distorted*.

Example 3. Consider again Example 2. Inequality $(1-x_0)+x_1+x_2+x_3 \geq 2$ removed all five points of the ball $\mathcal{B}(1,(1,0,0,0))$. Suppose now that we want to remove all the above points except from $(0,0,0,0)$ and $(1,0,1,0)$. Then, intuitively it is enough to increase a little bit the coefficient a_0 corresponding to point $(0,0,0,0)$, that is the coefficient of $(1-x_0)$, to be sure that when $x_0 = 0$ then $a_0(1-x_0) \geq 2$. As for the other points of the ball $x_0 = 1$, this change does not have any impact on them. In the same way, we can increase the coefficient a_2 before x_2, to be sure to keep $(1,0,1,0)$. One can check that $2(1-x_0) + x_1 + 2x_2 + x_3 \geq 2$ removes indeed the three remaining points of the ball.

The next proposition formalises the previous example to distorted balls of dimension d.

Proposition 3. *Let $\mathcal{B}(d,c) \subset \mathbb{F}_2^m$ be a ball of radius d from which we remove the set of points $\mathcal{Q} = (c \oplus \mathrm{Prec}(q)) \bigcap \mathcal{S}(d,c)$ for some $q \in \mathbb{F}_2^m$. Here $p \in \mathcal{Q}$ represents a possible transition towards the edge of the ball. We define $a \in \mathbb{Q}^m$ such that*

$$a_i = \begin{cases} \frac{d+1}{d} & \text{if } q_i = 1, \\ 1 & \text{otherwise.} \end{cases}$$

Then

$$\sum_{i=0}^{m-1} a_i\big[(1-c_i)x_i + c_i(1-x_i)\big] \geq d+1 \quad \Longleftrightarrow \quad x \notin \mathcal{B}(d,c) \backslash \mathcal{Q}.$$

Proof. First note that since $x, c \in \mathbb{F}_2^m$, $a_i\big[(1-c_i)x_i + c_i(1-x_i)\big] = a_i(x_i \oplus c_i)$ for all $i \in [0, m-1]$.

- If $x \notin \mathcal{B}(d,c)$, then $\sum_{i=0}^{m-1} a_i(x_i \oplus c_i) \geq \sum_{i=0}^{m-1} x_i \oplus c_i \geq \mathrm{wt}(x \oplus c) \geq d+1$.
- If $x \in \mathcal{B}(d-1,c)$, then $\sum_{i=0}^{m-1} a_i(x_i \oplus c_i) \leq \frac{d+1}{d}(d-1) \leq d$.
- Let $x \in \mathcal{S}(d,c)$.
  - If $x \in \mathcal{Q}$ then $x \oplus c \preceq q$ and $\sum_{i=0}^{m-1} a_i(x_i \oplus c_i) = \frac{d+1}{d}\mathrm{wt}(x \oplus c) = d+1$.
  - If $x \notin \mathcal{Q}$ then $x \oplus c \not\preceq q$ and there exists some j such that $x_j \neq c_j$, $q_j = 0$ and $a_j = 1$. Then

$$\begin{aligned} &\sum_{i=0}^{m-1} a_i(x_i \oplus c_i) = 1 + \sum_{i \neq j} a_i(x_i \oplus c_i) \\ &\leq 1 + \frac{d+1}{d} \sum_{i \neq j} x_i \oplus c_i = 1 + \frac{d+1}{d}(\mathrm{wt}(x \oplus c) - 1) = 1 + \frac{d+1}{d}(d-1) \\ &< d+1. \end{aligned}$$

□

Remark 1. For $d = 1$, Proposition 3 implies that for any ball of radius 1 and for any subset $\mathcal{Q}$ of points on its edge, it is possible to create an inequality that removes the points in the ball minus the set $\mathcal{Q}$. For $d > 1$ the situation is a little bit different, as removing some points from the edge can also remove other points that we would like to keep. Despite this, inequalities created this way are usually interesting as they can permit to remove different points together.

Example 4. Consider again the example of PRESENT and let $\mathcal{B}(1,c)$ be the ball centred at $c = [0,1] : \mathcal{B}(1,c) = \{[0,1], [0,0], [0,3], [0,5], [0,9], [1,1], [2,1], [4,1], [8,1]\}$. By looking at the corresponding DDT (Appendix B) one can see that all the points in $\mathcal{B}(1,c)$ correspond to impossible transitions, except $[0,0]$. Then, if $q = [0,1]$, we have that $\mathcal{Q} = \{[0,0]\}$ and

$$x_0 + x_1 + x_2 + x_3 + 2(1-y_0) + y_1 + y_2 + y_3 \geq 1$$

removes $\mathcal{B}(1,c) \setminus \mathcal{Q}$.

The inequalities provided up to now can remove points in a single ball. While this provides a simple and fast algorithm by itself by simply going through all balls into a DDT and writing down the corresponding inequalities, in practice we only used this method in combination with a more powerful method that we detail in the next section. This new method permits us to remove points belonging to the union of three different balls of radius 1. We call this process *merging*.

2.6.1 Merging three balls of radius $d = 1$

This method has similarities with the adding inequalities method used in Algorithm 1, except that we combine inequalities obtained with the distorted balls approach. We start by computing distorted balls of radius 1 centred on impossible transitions. However, simply adding inequalities obtained from neighbouring distorted balls often results in "bad" inequalities, in the sense that they do not discard many impossible transitions. To get a more efficient method, we found that when slightly changing one of the distorted balls, adding 2 to the right-hand side of the sum of the three inequalities gives a better inequality. We now present in detail how to get such an inequality.

Let $\mathcal{I}$ be the set of all impossible transitions through the Sbox and let a, b and c be three distinct points in $\mathcal{I}$ such that

- $\mathrm{wt}(a \oplus b) = \mathrm{wt}(a \oplus c) = 1$,
- $b \neq c$,
- $a \oplus b \oplus c \in \mathcal{I}$.

Let $\mathcal{B}(1, a), \mathcal{B}(1, b)$ and $\mathcal{B}(1, c)$ be the corresponding balls of radius 1. We denote by

$$P_a = \mathcal{B}(1, a) \setminus \mathcal{I} \subseteq \mathcal{S}(1, a) \qquad P_b = \mathcal{B}(1, b) \setminus \mathcal{I} \subseteq \mathcal{S}(1, b)$$
$$P_c = \mathcal{B}(1, c) \setminus \mathcal{I} \subseteq \mathcal{S}(1, c)$$

the possible transition points inside each ball. Finally, consider the sets

$$Q_1 = P_a \oplus a \oplus b \subseteq \mathcal{S}(1, b) \qquad Q_2 = P_a \oplus a \oplus c \subseteq \mathcal{S}(1, c)$$
$$Q_3 = P_b \oplus b \oplus c \subseteq \mathcal{S}(1, c)$$

and let $\mathcal{Q}$ denote $P_a \cup P_b \cup P_c \cup Q_1 \cup Q_2 \cup Q_3$.

With Proposition 3, one can compute the inequalities $A(x) \geq 2, B(x) \geq 2$ and $C(x) \geq 2$ that remove respectively $\mathcal{B}(1, a) \setminus (\mathcal{Q} \cup \{c\}), \mathcal{B}(1, b) \setminus \mathcal{Q}$ and $\mathcal{B}(1, c) \setminus \mathcal{Q}$. With the above notations, we have the following proposition.

Proposition 4. *Let $x \in \{0, 1\}^m$, then*

$$A(x) + B(x) + C(x) \geq 8 \quad \Leftrightarrow \quad x \notin \big(\mathcal{B}(1, a) \cup \mathcal{B}(1, b) \cup \mathcal{B}(1, c)\big) \setminus \mathcal{Q}.$$

A proof for Proposition 4 and an example on the Sbox of PRESENT is provided in Appendix C. The method is summarized in Algorithm 3.

The sets R_a, R_b and R_c in Algorithm 3 correspond to the points inside each ball that have to be kept when writing down the equation for the corresponding distorted ball. More precisely, $R_a = P_a \cup \{c\}, R_b = P_b \cup Q_1$ and $R_c = P_c \cup Q_2 \cup Q_3$. It is interesting to note that there is no symmetry between b and c: if one changes the roles of b and c, then the new inequality created will remove a different subset of points from the three balls, giving thus a different inequality for our collection.

We applied Alg. 3 together with Alg. 2 and Proposition 3 to create a large set of inequalities for the Sboxes of SKINNY-8 and AES and applied a MILP minimization problem to find a small set of inequalities to represent each Sbox. The results can be visualized in Table 4. The resulting MILP problem took only a few seconds to terminate for SKINNY-8 while the optimization could not be terminated for AES. Even though, the number of inequalities that we got by stopping the optimization process after a few days of computation provided us with a much lower number than the best previous result. Of course, by pushing the optimization further, it is possible to get even less inequalities for AES but one has to remember that obtaining the absolute minimum does not usually lead to the quickest solving time.

Last, we also applied the combination of the three above methods (Alg. 2, Alg. 3 and Proposition 3) to all 4-bit Sboxes of Table 2 and for all of them we obtained the same number of inequalities as with Algorithm 1.

Algorithm 3 Create new inequalities from all possible triples of distorted balls

procedure AddThreeBalls($\mathcal{I}$)
  $\mathcal{C} \leftarrow \emptyset$ ▷ Initialize the set of inequalities.
  for all $(a, b, c) \in \mathcal{I}$ s. t. $\mathrm{wt}(a \oplus b) = \mathrm{wt}(a \oplus c) = 1$ and $b \neq c$ **do**
    $P_a \leftarrow \mathcal{B}(1, a) \setminus \mathcal{I}$ $P_b \leftarrow \mathcal{B}(1, b) \setminus \mathcal{I}$
    $R_a \leftarrow P_a \cup \{c\}$, $R_b \leftarrow P_b$, $R_c \leftarrow \mathcal{B}(1, c) \setminus \mathcal{I}$
    for all $p \in P_a$ **do**
      $R_b \leftarrow R_b \cup \{p \oplus a \oplus b\}$
      $R_c \leftarrow R_c \cup \{p \oplus a \oplus c\}$
    end for
    for all $p \in P_b$ **do**
      $R_c \leftarrow R_c \cup \{p \oplus b \oplus c\}$
    end for
    Use Proposition 3 to write down an inequality $A(x) \geq 2$ removing $\mathcal{B}(1, a) \setminus R_a$
    Use Proposition 3 to write down an inequality $B(x) \geq 2$ removing $\mathcal{B}(1, b) \setminus R_b$
    Use Proposition 3 to write down an inequality $C(x) \geq 2$ removing $\mathcal{B}(1, c) \setminus R_c$
    $C_{new} \leftarrow A(x) + B(x) + C(x) \geq 8$ ▷ New interesting inequality
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_{new}\}$
  end for
end procedure

Table 4: Number of inequalities to model the corresponding Sboxes, where the set of initial inequalities was generated by three different methods: Alg. 2 and 3 and Prop. 3

	# Inequalities	
Sbox	[AST+17]	Alg. 2 and 3 and Prop. 3.
Skinny-8	372	**302**
AES	8302	**2882**

2.7 Comparing different techniques for Sbox modeling

Using 5 or 6-bit Sboxes inside a cipher is less common than using 4-bit and 8-bit ones. However, some designs, e.g. Keccak, Ascon or Fides among others, use such Sboxes for different reasons each: Good masking properties or optimal resistance against differential cryptanalysis among others. Indeed, APN permutations, that is permutations whose DDTs are only composed of 0s and 2s and that have the best possible differential properties, exist for 5 and 6 bits. We tested our algorithms on some Sboxes of this size, mainly for permitting comparison between the different developed methods. Indeed, these sizes, not as small as 4 bits and not as large as 8 bits are ideal for permitting all of the algorithms to run (even the ones based on the computation of the convex hull) and for providing non-trivial comparisons. The results are summarized in Table 5. The first method, that we note as Convex Hull, corresponds to the method based on the H-representation of the convex hull provided by Sun et al. in [SHW+14a].

As one can see from the above table, Algorithm 1 and the combination of Algorithm 2, Algorithm 3 and Proposition 3 are the ones giving always the best results. Algorithm 1 is almost always better but it has the disadvantage that it cannot be applied to larger dimensions.

Table 5: Number of inequalities for modeling various 5 and 6-bit Sboxes with four different methods.

			# Inequalities			
n	Sbox	Citation	Convex Hull	Alg. 2	Alg. 1	Alg. 2 and 3 and Prop. 3
5	Keccak	[BDPA09]	46	46	**34**	36
	Ascon	[DEMS19]	40	59	**32**	49
	Fides-5	[BBK+13]	79	124	64	**61**
	SC2000-5	[SYY+01]	82	123	66	**64**
6	Apn-6 †	[BDMW10]	195	288	**167**	179
	Fides-6	[BBK+13]	223	455	**180**	194
	SC2000-6	[SYY+01]	241	567	218	**214**

†The concrete function analyzed here is the one given through its table representation by John Dillon in his talk at *Fq09* [Dil09].

3 Linear layer modeling

As seen in the previous section, modeling in the context of MILP valid propagations through 8-bit Sboxes may lead to large systems of $\mathbb{R}$-linear inequalities. One would think that modeling a linear layer is much easier. While this is true for simple linear layers as the ones of Present or Gift that consist in simple bit-permutations, modeling other linear layers can also lead to large systems of $\mathbb{R}$-linear inequalities. This is in particular due to the difficulty of efficiently modeling the `XOR` operation that is the major component of most diffusion layers. This is explained in the next subsection.

3.1 `XOR` modeling

Modeling a linear layer is often related to how the `XOR` operation is modeled. The following proposition gives an idea of how difficult it can be to efficiently model a linear layer without dummy variables.

Proposition 5. *The equation* $x_0 \oplus x_1 \oplus \ldots \oplus x_{n-1} = 0$ *needs at least* 2^{n-1} $\mathbb{R}$*-linear inequalities of the form*

$$\sum_{i=0}^{n-1} c_i x_i + d \geq 0, \quad c \in \mathbb{R}^n, d \in \mathbb{R}$$

to characterise the set of its solutions in $\mathbb{F}_2^n$.

Proof. First, we can exhibit such a set of inequalities. Indeed, for each $a \in \mathbb{F}_2^n$ such that $a_0 \oplus a_1 \oplus \ldots \oplus a_{n-1} = 1$, we have seen in Section 2 that we can write down an inequality that only eliminates a from the set of possible solutions. Since there are exactly 2^{n-1} such points, we have 2^{n-1} inequalities modeling $\left\{ x \in \mathbb{F}_2^n \,\middle|\, x_0 \oplus \ldots \oplus x_{n-1} = 0 \right\}$.

Let us suppose now that there exists an inequality $c \cdot x + d \geq 0$ that eliminates at the same time two points a and b such that $a_i = 0$ and $b_i = 1$ for some $i \in [0, n-1]$. Let e be the vector $(0, \ldots, 1, \ldots, 0)$ with $e_i = 1$. Then,

- if $c_i \leq 0$, $c \cdot (a \oplus e) + d = c \cdot a + d + c_i < 0$.
- Otherwise, if $c_i > 0$, $c \cdot (b \oplus e) + d = c \cdot b + d - c_i < 0$.

This means that if an inequality eliminates two different points on the cube $\{0,1\}^n$, it necessarily also eliminates two points with Hamming distance 1.

Moreover, two points a and b such that $\mathrm{wt}(a \oplus b) = 1$ cannot both be solutions of $x_1 \oplus \ldots \oplus x_n = 0$. This explains why we need as many $\mathbb{R}$-linear inequalities as points to eliminate, i.e. 2^{n-1}. □

Therefore, a naïve way to model a linear layer is to model each `XOR` operation in the way showed in Proposition 5. We will often refer to this as the *naïve method*. The rest of this section is dedicated to the presentation of more efficient ways for modeling general linear layers.

3.2 General modeling

When modeling a mathematical operation for MILP with a system of linear inequalities, the input and output variables play the same role inside each inequality. This shows that modeling a matrix M means modeling the kernel of $A = (M|I)$, where I is the identity matrix. Indeed, for any matrix M with entries in $\mathbb{F}_2$,

$$Mx = y \Leftrightarrow Mx \oplus y = 0 \Leftrightarrow A \begin{pmatrix} x \\ y \end{pmatrix} = 0.$$

One can then model the equation given by each row of A with the `XOR` modeling. But as we have just seen, the number of constraints for modeling one `XOR` operation grows exponentially with the number of involved variables. Since our goal is to model the kernel of A and since it is known that for any invertible matrix $P \in \mathrm{GL}_n(\mathbb{F}_2), \mathrm{Ker}(P \cdot A) = \mathrm{Ker}\, A$, the idea is to find a matrix $P \in \mathrm{GL}_n(\mathbb{F}_2)$ such that the rows of $P \cdot A$ have minimum Hamming weight and induce thus a minimal number of `XOR` operations. More precisely, this means finding an invertible matrix P that minimizes

$$\sum_{i=1}^{n} 2^{\mathrm{wt}(P \cdot A)_{i,\star} - 1}, \tag{4}$$

where $\mathrm{wt}(P \cdot A)_{i,\star}$ corresponds to the Hamming weight of the i-th row of $P \cdot A$. The question is now how to find such a matrix P. A first idea would be to search for minimum weight codewords in the linear code generated by the rows of A, as this is done when computing the linear branch number of the matrix M. For example, consider the matrix

$$M_{\text{Midori}} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad (M_{\text{Midori}}|I) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

used in the linear layer of Midori. This matrix has branch number 4 which means that for all $x \in \mathbb{F}_2^4$, $\mathrm{wt}(x \cdot (M|I)) \geq 4$. Hence we cannot hope for a more minimal modeling of this linear operation based on the `XOR` modeling than the one given by $(M|I)$. However, let us take the example of the Skinny `MixColumns` operation given by the matrix M_{Skinny} below. In that case, the code generated by $(M_{\text{Skinny}}|I)$ has minimum distance 2 and it is equivalently generated by the matrix A_{Skinny} obtained by adding in $\mathbb{F}_2$ the fourth line to the first line of $(M|I)$.

$$M_{\text{Skinny}} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad A_{\text{Skinny}} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

While the naïve `XOR` modeling of $(M_{\text{Skinny}}|I)$ would have needed $2^3 + 2 + 2^2 + 2^2 = 18$ inequalities, using the above matrix for the `XOR` modeling only requires 14 inequalities.

To demonstrate that this representation is more efficient in practice compared to the naïve approach, we computed the time it takes for the `Gurobi` Optimizer [GO20] to reach the minimum number of active Sboxes over several rounds of Skinny-128 for the two different modelings of `MixColumns`. In this experiment, in order to emphasize the impact

of the linear layer modeling and to avoid correlations with the modelings of the other parts of the cipher, we used a very simple modeling for the 8-bit Sbox. This Sbox modeling, introduced in [AST+17] under the name of *arbitrary Sbox mode*, needs only $2n$ inequalities for the whole Sbox but only models the following behaviour: if at least one input bit is active then at least one output bit has to be active and vice versa. The timings (in seconds) of this experiment can be visualised in Table 6. We emphasize that the reported timings is the time for the solver to *reach* what we already know to be the minimum number of active Sboxes. Indeed, as even with the improved modeling the number of inequalities still remains high, our MILP solver takes too long for terminating and thus *proving* that the found upper bound on the minimum number of active Sboxes is tight. The minimum number of active Sboxes for Skinny has been computed by its designers in [BJK+16] thanks to wordwise modelings.

Table 6: Computation time in seconds for the `Gurobi` solver to find the minimum number of active Sboxes over r rounds for Skinny-128 with two different modelings for `MixColumns`.

Number of rounds	6	7	8	9	10
Minimum number of active Sboxes	16	26	36	41	46
Time to reach this minimum with the new modeling	0	0	0	16	5200
Lowest upper bound with naïve linear layer or - if the minimum was reached	-	-	-	-	47
Time at which we stopped the experiment or time after which the minimum was reached	16	35	30	1862	14600

It is obvious from this table that the new modeling of the Skinny linear layer reduces importantly the solving time. We propose now a new algorithm, Algorithm 4, that given the matrix $A = (M|I)$, finds a matrix P that minimizes Eq. (4). First, the matrix P is initialized to the identity matrix. Then, the algorithm proceeds in a row-wise manner and searches at each step to replace the current row with a better one. To start with, it searches to replace the first row of A with a codeword of the form

$$m \cdot A, \ m \in \left\{ x \in \mathbb{F}_2^n \,\middle|\, x_1 = 1 \right\} \text{ and } \mathrm{wt}(m \cdot A) < \mathrm{wt}(A_{1,\star}).$$

After this first step, the matrix P is updated as

$$P = \begin{pmatrix} 1 & m_2 & \cdots & m_n \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}.$$

The algorithm then searches for a replacement for the second row of the matrix $P \cdot A$ in the same way and updates the matrix P if a lower weight codeword has been found.

Algorithm 4, whose time complexity is $n2^{n-1}$ multiplications of a n-bit binary vector with a matrix with n rows and an arbitrary number of N columns, finds notably the previous result for Skinny (14 inequalities). We also applied Algorithm 4 to the Aes `MixColumns`. The naïve `XOR` modeling gives 2176 inequalities for this matrix and Algorithm 4 does not improve this quantity.

We develop now a different idea that permits in some cases to significantly improve the number of inequalities needed for modeling the linear layer, compared to both the naïve approach and Algorithm 4.

Algorithm 4 Given a binary matrix A of size $n \times N$ returns $P \in \mathrm{GL}_n(\mathbb{F}_2)$ minimizing Eq. (4).

procedure FINDP(A)
  $A^{\mathrm{mut}} \leftarrow A$
  $P \leftarrow I_n$
  for $\ell \in \{1, \ldots, n\}$ **do**
    $m_{\mathrm{best}} \leftarrow A^{\mathrm{mut}}_{\ell}$
    for all $m \in \left\{ x \in \mathbb{F}_2^n \,\middle|\, x_\ell = 1 \right\}$ **do**
      if $\mathrm{wt}(m \cdot A^{\mathrm{mut}}) < \mathrm{wt}(m_{\mathrm{best}}) \cdot A^{\mathrm{mut}}$ **then**
        $m_{\mathrm{best}} \leftarrow m$
      end if
    end for
    $A^{\mathrm{mut}}_{\ell} \leftarrow m_{\mathrm{best}} \cdot A^{\mathrm{mut}}$
    $P \leftarrow \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & m_{\mathrm{best}} & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} \cdot P$
  end for
  return P
end procedure

3.3 Changing the Sbox modeling for improving the linear one

The idea of this approach consists in changing the Sbox modeling. Indeed, if we find an invertible block-diagonal matrix (with blocks having the size of the Sbox)

$$Q = \begin{pmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_{2b} \end{pmatrix}$$

where b is the number of words on which `MixColumns` operates (e.g. $b = 4$ for the AES, SKINNY or MIDORI), then changing the modeling of the Sbox S into the modelings of $Q_i^{-1} \circ S \circ Q_{i+b}^{-1}$ for all $i \in [1, b]$ allows for a new, potentially better `XOR` modeling of the `MixColumns` operation with the equation $(M|I)Q = 0$. Once a convenient matrix Q has been found, modeling $(M|I)Q = 0$ can be done using Algorithm 4. In summary, with this method, the problem of finding a good `XOR` modeling for the linear layer boils down to finding a matrix $P \in \mathrm{GL}_n(\mathbb{F}_2)$ and a block-diagonal matrix $Q \in \mathrm{GL}_{2n}(\mathbb{F}_2)$ such that $P(M|I)Q$ minimizes (4).

We propose Algorithm 5 for finding such matrices P and Q. The idea of this algorithm is to iterate alternating searches for P and Q, hoping that modifying one of P or Q will allow further optimization. This algorithm then takes as parameter the number of desired iterations $p \in \mathbb{N}$. In practice however, for all of our experiments, we have never needed $p \geq 2$.

Applying Algorithm 5 on the AES `MixColumns` operation with parameter $p = 1$ gives P and Q such that the quantity of Equation (4) initially at 2176 drops down to 1088. However, Algorithm 5 does not give better results for MIDORI or Algorithm 4 for SKINNY. Matrices P and Q for the AES are given in Appendix E.

To measure at which point such a change in the modeling of the linear layer can be an improvement for the running time, we ran an experiment for the AES similar to the

Algorithm 5 Given a matrix A of size $n \times N$, find $P \in \mathrm{GL}_n(\mathbb{F}_2)$ and block-diagonal Q with $2b$ blocks such that (P, Q) minimizes $\sum_{i=1}^{n} \mathrm{wt}((P \cdot A \cdot Q)_i)$.

1: **procedure** FINDB(A, p, b)
2: $P \leftarrow$ FINDP(A)
3: $A^{\mathrm{mut}} \leftarrow PA^{\mathrm{mut}}$
4: $Q \leftarrow I_{2n}$
5: **loop** p times
6: **for all** $i \in [1, 2b]$ **do**
7: $C_i \leftarrow$ columns $\frac{n \cdot i}{b}, \cdots, \frac{n \cdot (i+1)}{b} - 1$ of A^{mut}
8: $Q_i \leftarrow \mathrm{FINDP}(C_i^T)^T$
9: **end for**
10: $Q \leftarrow Q \cdot \begin{pmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_{2b} \end{pmatrix}$
11: $A^{\mathrm{mut}} \leftarrow A^{\mathrm{mut}} \cdot \begin{pmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_{2b} \end{pmatrix}$
12: $P \leftarrow \mathrm{FINDP}(A^{\mathrm{mut}}) \cdot P$
13: $A^{\mathrm{mut}} \leftarrow PA^{\mathrm{mut}}$
14: **end loop**
15: **return** P, Q
16: **end procedure**

Table 7: Time in seconds to reach the lowest upper bound for the minimum number of active Sboxes for the AES.

Number of rounds	5	6	7	8
Lowest upper bound with improved linear layer	28	35	47	65
Time needed to reach this upper bound	105	117	5547	6900
Lowest upper bound with naïve linear layer	31	40	49	69
Time at which we stopped the experiment	2300	2200	8200	10000

one above for Skinny-128. One important difference is that wordwise modelings for the AES have not been able to provide tight lower bounds for the minimum number of active Sboxes because of byte multiplications needed in the `MixColumns` operation. For example in [MWGP11], the authors used the branch number of the `MixColumns` operation to obtain lower bounds. In Table 7, we hence give the lowest upper bound on the minimum number of active Sboxes reached with the improved linear layer modeling after the given number of seconds and the same upper bound reached by the naïve linear layer after the given number of seconds.

For a discussion on the complexity of Algorithms 4 and 5, see Appendix D.

3.3.1 Modeling of affine equivalent Sboxes

As seen in this section, to apply Algorithm 5, instead of modeling the cipher's original Sbox S, one will need to model one or more Sboxes that are linearly equivalent to S. Therefore, it is necessary to ensure that the modeling of these linearly equivalent Sboxes is not (much) worse than the modeling for S. This leads to the following more general question: "How does the modeling of an Sbox gets affected by affine equivalence?" In our experiments with AES, the only needed affine equivalent Sbox — $\texttt{Sbox}_{\text{AES}} \circ Q_0$ where Q_0 is given in Appendix E — had a very similar modeling to the original one $\texttt{Sbox}_{\text{AES}}$, leading notably to almost the same number of final inequalities. It is however not clear for us what happens in the general case and we believe that this constitutes an interesting open problem. For example, it could be useful to know for any given Sbox whether one can compute an affine equivalent Sbox that can be modeled with much less inequalities. A related question is whether a lower bound on the number of needed inequalities in the modeling can be found accross the equivalence class of an Sbox.

3.4 Other applications

Besides AES, Skinny and Midori we also applied Algorithms 4 and 5 to the linear layers of some other block ciphers. The obtained results are summarized in Table 8.

4 Impact of the new modelings on the solving time

In this section we analyze the impact of our new modelings on the running time of a MILP optimization problem. We chose to perform our experiments on the problem of deciding whether a differential $(\delta_{in}, \delta_{out})$ is possible, which in practice consists in finding a possible differential characteristic, if this one exists. This kind of computation is used in practice in impossible differential cryptanalysis. We will provide more details on the full problem in Section 5.

The goal of these experiments is to quantify in terms of time the impact of both the Sbox and the `MixColumns` modelings. For this, we considered two different modelings for the Sbox: the logical condition modeling method of Section 2.4 and a combination of the

Table 8: Number of inequalities to model a diffusion matrix for MILP without dummy variables with the naïve way, and with the two algorithms developed in this section. A description of the linear layers of the three last ciphers can be found in Appendix F.

	# Inequalities		
Cipher	Naïve	Algorithm 4	Algorithm 5
Midori-128	**32**	32	32
Skinny-128	18	**14**	14
AES	2176	2176	**1088**
Aria	**2048**	2048	2048
Anubis	7168	2032	**1680**
Saturnin	5632	**352**	352

Table 9: Computation time (sec) for 32 rounds of Skinny and 5 rounds of AES.

			`Sbox` modeling	
Cipher	Sample	`MixColumns` modeling	Alg. 2	Alg. 2 + Alg. 3
Skinny	1975 pairs	Naïve	185	172
		Improved	**21**	29
AES	Dense (4594 pairs)	Naïve	43	**16**
		Improved	42	25
	Sparse (259 pairs)	Naïve	1600	**441**
		Improved	1944	662

methods of Sections 2.4 and 2.5. For the linear layer we also considered two cases: in the first case we model the `MixColumns` matrix via the naïve `XOR` modeling and in the second we model it with the improved `XOR` modeling given by Algorithms 4 and 5. We then considered all four different combinations of these Sbox and `MixColumns` modelings. For launching the experiment, we chose a random set of input and output difference pairs $(\delta_{in}, \delta_{out})$ and asked `Gurobi` to verify whether the differential $(\delta_{in}, \delta_{out})$ is possible.

We ran this experiment for Skinny-128 and the AES and present in Table 9 the average computation times for each cipher. Quantiles and other statistics are available in Appendix G. For the AES, the computation time is dramatically higher when the input and output are sparse (have much more zeros than ones) than when they are dense, hence the need for two different tables. Our intuition for this fact is that there are many more possible differential characteristics inside the differential when the input and the output are dense, which makes it easy for the solver to find one in a few seconds. For the application on AES with sparse ends, we also indicate to our models which bytes are active and inactive in the first two and last two rounds. This reduces the running time of the solver by approximately a factor 10.

As we can see, for Skinny-128 the change of modeling for the linear layer has a much bigger impact on the running time than the change of the Sbox modeling. There are in our opinion three possible reasons for that: First, the number of rounds under study for Skinny, 32, is high enough for the modeling of the linear layers to have impact. Then, as one can see from Table 4, the difference between the two Sbox modelings is not significant. Finally, the number of inequalities per Sbox is rather low (< 400), which gives the linear layer modeling more impact as well.

For the AES, the inverse phenomenon happens: the linear layer modeling does not have an important impact whereas the Sbox modeling divides the running time by 2. Again, we think that a possible explanation of this is that the number of rounds, 5, is too low for the

linear layer to a have an important impact, the difference between the two Sbox modelings is significant (see Table 4) and the number of inequalities per Sbox is rather high (≈ 3000) making that the Sbox computations have a higher proportion in the global process.

5 Applications on impossible differential cryptanalysis

Impossible differential cryptanalysis is a powerful and well-studied cryptanalytic technique introduced independently by Knudsen [Knu98] and Biham et al. [BBS99]. Its principle consists in finding impossible differentials, i.e. an input difference δ_{in} and an output difference δ_{out} that cannot be connected. Any impossible differential attack starts with the discovery of an impossible differential $(\delta_{in}, \delta_{out})$ covering a maximal number of rounds. Traditionally this was done with the $\mathcal{U}$-method [KHS+03] or its extensions [LLWG14, WW12]. However, in 2017, Sasaki and Todo [ST17b], showed that MILP can be successfully used to prove resistance against impossible differential attacks or for discovering new impossible differential distinguishers. For proving (partial) resistance against impossible differential cryptanalysis with MILP, and as briefly explained in Section 4, one can choose a set of input and output pairs and then solve a MILP differential propagation problem with the given cipher model for each of those pairs. The chosen set is typically composed of all possible inputs and outputs with exactly one active byte, i.e. for which exactly one byte has a difference. When all of those computations result in a valid differential transition, showing thus that the input and output differences can be connected, we consider that resistance against impossible differential cryptanalysis has been partially proven, where partially applies to the fact that the input and output spaces are restricted. However, as seen in Section 4, a single computation with a fixed input and output difference can be too long for permitting to do such kind of proofs for a large meaningful enough set of input/output pairs. To overcome this problem Sasaki and Todo introduced in Section 5 of [ST17b], the *Differential Possibility Equivalence technique*. This technique reduced the number of MILP instances to solve and permitted to drastically reduce the overall running time of this process. In what follows we briefly present this technique. Then we show how to further improve this method for decreasing the running time further. We applied this technique to provide partial resistance against differential cryptanalysis for 5-round AES and 13-round Skinny-128.

5.1 The Differential Possibility Equivalence technique

We consider for the rest of this section that an R-round MILP model is composed of R non-linear layers and begins and ends with a non-linear layer. It is then also obviously composed of $R-1$ linear layers.

Suppose that we search for R-round impossible differentials for an SPN cipher. Suppose further that the pairs of input and output differences are restricted in a set $\mathcal{S}$ and that we have computed a possible differential characteristic for R rounds $x_0 \xrightarrow{R} y_0$. The idea of the Differential Possibility Equivalence technique is to exploit this possible differential characteristic to discard other possible transitions $(x, y) \in \mathcal{S}$ without computing at each time a new R-round differential characteristic $x \xrightarrow{R} y$ from scratch. To do so, one chooses r_{in} and r_{out} such that $r_{\text{in}} + r_{\text{out}} < R$ and gets the differences x' and y' in the already computed path

$$x_0 \xrightarrow{r_{\text{in}}} x' \xrightarrow{R-r_{\text{in}}-r_{\text{out}}} y' \xrightarrow{r_{\text{out}}} y_0.$$

One then discards all $(x, y) \in \mathcal{S}$ such that the transitions $x \xrightarrow{r_{\text{in}}} x'$ and $y' \xrightarrow{r_{\text{out}}} y$ are possible, which means computing two smaller differential paths on r_{in} and r_{out} rounds. Sasaki and Todo introduced this technique in [ST17b] with x' and y' respectively right before and right after the first and last non-linear layers (hence with $r_{\text{in}} = r_{\text{out}} = 1$),

finding x and y by directly looking at the DDT of the Sboxes used in those non-linear layers. However, it can be interesting to try the same with other values for r_{in} and r_{out}, using MILP models for checking differential paths $x \xrightarrow{r_{\text{in}}} x'$ and $y' \xrightarrow{r_{\text{out}}} y$.

The choice of r_{in} and r_{out} for finding a minimal running time depends on the cipher and the running times for computing differential characteristics for R, r_{in} and r_{out} respectively rounds. Moreover, one should avoid to check differential transitions $x \xrightarrow{r_{\text{in}}} x'$ and $y' \xrightarrow{r_{\text{out}}} y$ when they have a low probability to be possible. In the search for impossible differentials with exactly one active input and one active output byte, it appears that it is rather efficient to restrain the checks for (x, y) such that x shares the same active byte as x_0 and y shares the same active byte as y_0. Indeed, for the AES with $R = 5, r_{\text{in}} = r_{\text{out}} = 2$, the computation of one R-round differential characteristic $x_0 \xrightarrow{R} y_0$ allows to discard all the other pairs (x, y) with the same active bytes. For SKINNY-128 with $R = 13, r_{\text{in}} = r_{\text{out}} = 2$, for 2 checks $x \xrightarrow{2} x'$ or $y' \xrightarrow{2} y$, one possible transition gets discarded on average.

5.2 Applications to Skinny-128 and AES

In [ST17b] the authors used MILP for proving the maximal number of rounds for which impossible differentials exist for many different designs, if the input and output differences are restricted inside one word (bit, nibble or byte, depending on the design). Most of the analyzed ciphers are based on 4-bit Sboxes while for the analyzed designs using 8-bit Sboxes the details of the Sboxes are not taken into account. The only exception is MIDORI-128 but whose Sboxes are constructed from two 4-bit ones.

Our new modelings for both big Sboxes and linear layers and the above generalization of the Differential Possibility Equivalence technique permitted us to complement the work done in [ST17b] for 8-bit based ciphers, by taking in particular the Sbox details into account. For both 5-round AES and 13-round SKINNY-128 [1], for each pair of input/output bytes (i, j), we ran our models with the Differential Possibility Equivalence technique on the set with 255×255 elements with byte i active in input and byte j active in output. Those computations provide us with proofs for partial resistance against impossible differential cryptanalysis.

Skinny-128. For the parameters $R = 13, r_{\text{in}} = r_{\text{out}} = 2$, the proof for each pair of input/output active byte has been completed in an average time of 15 minutes on 2 cores of Intel XEON E3-1240 v5. To the best of our knowledge, this proof is a new result

AES. For the parameters $R = 5, r_{\text{in}} = r_{\text{out}} = 2$, the proof for each pair of input/output active byte completed in an average time of 10 hours on 2 cores of Intel XEON E3-1240 v5. Sun et al. proved in [SLG+16] without any restriction on the number of active input or output bytes, that there are no 5-round impossible differentials for AES, unless the details of the Sbox are taken into account. By taking the details of the Sbox into account, we provide an extension of Sun et al.'s proof, in the case of one active input and one active output byte. Note however that Wang and Jin gave recently in [WJ19] a theoretical proof for 5 rounds of AES. Indeed, they proved, that for any number of active input/output bytes, all differentials are possible if round-keys are taken uniformly at random.

6 Conclusion

In this paper, we presented new techniques for improving MILP models simulating differential properties through Sboxes and `MixColumns` operations. We found better modelings

[1] There is an ambiguity in the literature over the number of rounds on which impossible differentials are found for SKINNY. For us, 13 rounds means that there are 13 Sbox layers and 12 linear layers.

for various Sboxes with sizes ranging from 4 to 8 bits with three different approaches: adding inequalities given by the H-representation of the convex hull of possible transitions, packing impossible transitions in affine subspaces of the form $a \oplus \mathrm{Prec}(u)$ or packing them in (possibly "merged") distorted balls. Our techniques for Sbox modeling are very general: they basically give efficient algorithms to represent any subset of $\{0,1\}^n$ with a small number of inequalities. Those techniques could then have applications beyond differential cryptanalysis. We also introduced a link between the modeling of $\mathbb{F}_2$-linear operations and the search for a basis with minimal weight codewords inside $\mathbb{F}_2$-linear codes. We gave algorithms based on this link to reduce the inequality cost of `MixColumns` modelings.

We then gave insights on how those techniques impact the performance of computing a differential characteristic with a fixed input and output for Skinny-128 and the AES. Those two cases respectively demonstrate the benefits of better modelings for the `MixColumns` operations and for the Sboxes. Finally, we applied those techniques to proving partial resistance of 13-round Skinny and 5-round AES against impossible differential cryptanalysis with the specific properties of the Sboxes fully taken into account.

Lastly, our techniques could be applied in a straightforward manner to the search for best differential characteristics in various ciphers as explained in [AST+17] and to similar problems of linear cryptanalysis. The optimal way of solving *Problem 2* to improve performance remains an open problem.

References

[aes01] Federal Information Processing Standards Publication (FIPS 197). Advanced Encryption Standard (AES), 2001.

[AST+17] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP Modeling for (Large) S-boxes to Optimize Probability of Differential Characteristics. *IACR Trans. Symmetric Cryptol.*, 2017(4):99–129, 2017.

[BBK+13] Begül Bilgin, Andrey Bogdanov, Miroslav Knezevic, Florian Mendel, and Qingju Wang. Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In *CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 142–158. Springer, 2013.

[BBS99] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 12–23. Springer, 1999.

[BDMW10] K.A. Browning, J.F. Dillon, M.T. McQuistan, and A.J. Wolfe. An apn permutation in dimension six. *Finite Fields: theory and applications*, 42:33–42, 2010.

[BDPA09] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. http://keccak.noekeon.org/, 2009.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 123–153, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 321–345. Springer, 2017.

[BSVMMH84] Robert King Brayton, Alberto L. Sangiovanni-Vincentelli, Curtis T. McMullen, and Gary D. Hachtel. Logic Minimization Algorithms for VLSI Synthesis. *Kluwer Academic Publishers*, 1984.

[CDL+19] Anne Canteaut, Sébastien Duval, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Thomas Pornin, and André Schrottenloher. Saturnin: a suite of lightweight symmetric algorithms for post-quantum security, 2019. Candidate to the Lightweight Cryptography NIST Competition.

[DEMS19] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to Round 1 of the NIST Lightweight Cryptography project, 2019.

[Dil09] John Dillon. APN polynomials: An Update. Invited talk at Fq9, the 9th International Conference on Finite Fields and Applications, Dublin, July 13-17, 2009.

[GO20] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[KHS+03] Jongsung Kim, Seokhie Hong, Jaechul Sung, Changhoon Lee, and Sangjin Lee. Impossible differential cryptanalysis for block cipher structures. In *INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 82–96. Springer, 2003.

[KKP+03] Daesung Kwon, Jaesung Kim, Sangwoo Park, Soo Hak Sung, Yaekwon Sohn, Jung Hwan Song, Yongjin Yeom, E-Joong Yoon, Sangjin Lee, Jaewon Lee, Seongtaek Chee, Daewan Han, and Jin Hong. New block cipher: ARIA. In *ICISC 2003*, volume 2971 of *Lecture Notes in Computer Science*, pages 432–445. Springer, 2003.

[KLPR10] Lars R. Knudsen, Gregor Leander, Axel Poschmann, and Matthew J. B. Robshaw. PRINTcipher: A Block Cipher for IC-Printing. In *CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 16–32. Springer, 2010.

[Knu98] L. R. Knudsen. DEAL – A 128-bit cipher. Technical Report, Department of Informatics, University of Bergen, Norway, 1998.

[LLWG14] Yiyuan Luo, Xuejia Lai, Zhongming Wu, and Guang Gong. A unified method for finding impossible differentials of block cipher structures. *Inf. Sci.*, 263:211–220, 2014.

[McC56] Edward J. McCluskey. Minimization of Boolean functions. *The Bell SystemTechnical Journal*, 35(6):1417—-1444, 1956.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In *Inscrypt 2011*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.

[Qui52] Willard Van Orman Quine. The Problem of Simplifying Truth Functions. *The American Mathematical Monthly*, 59(8):521—-531, 1952.

[Qui55] Willard Van Orman Quine. A Way to Simplify Truth Functions. *The American Mathematical Monthly*, 62(9):627—-631, 1955.

[SHW+14a] Siwei Sun, Lei Hu, Meiqin Wang, Peng Wang, Kexin Qiao, Xiaoshuang Ma, Danping Shi, Ling Song, and Kai Fu. Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Cryptology ePrint Archive, Report 2014/747, 2014. https://eprint.iacr.org/2014/747.

[SHW+14b] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. In *ASIACRYPT 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.

[SLG+16] Bing Sun, Meicheng Liu, Jian Guo, Vincent Rijmen, and Ruilin Li. Provable security evaluation of structures against impossible differential and zero correlation linear cryptanalysis. In *EUROCRYPT 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 196–213. Springer, 2016.

[ST17a] Yu Sasaki and Yosuke Todo. New Algorithm for Modeling S-box in MILP Based Differential and Division Trail Search. In *SecITC 2017*, volume 10543 of *Lecture Notes in Computer Science*, pages 150–165. Springer, 2017.

[ST17b] Yu Sasaki and Yosuke Todo. New Impossible Differential Search Tool from Design and Cryptanalysis Aspects - Revealing Structural Properties of Several Ciphers. In *EUROCRYPT 2017*, volume 10212 of *Lecture Notes in Computer Science*, pages 185–215, 2017.

[SYY+01] Takeshi Shimoyama, Hitoshi Yanami, Kazuhiro Yokoyama, Masahiko Takenaka, Kouichi Itoh, Jun Yajima, Naoya Torii, and Hidema Tanaka. The block cipher SC2000. In *FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 312–327. Springer, 2001.

[The20] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.1)*, 2020. `https://www.sagemath.org`.

[Wag99] David Wagner. The boomerang attack. In Lars Knudsen, editor, *Fast Software Encryption*, pages 156–170, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[WJ19] Qian Wang and Chenhui Jin. More accurate results on the provable security of AES against impossible differential cryptanalysis. *Des. Codes Cryptogr.*, 87(12):3001–3018, 2019.

[WW11] Shengbao Wu and Mingsheng Wang. Security evaluation against differential cryptanalysis for block cipher structures. *IACR Cryptol. ePrint Arch.*, 2011:551, 2011.

[WW12] Shengbao Wu and Mingsheng Wang. Automatic search of truncated impossible differentials for word-oriented block ciphers. In *INDOCRYPT 2012*, volume 7668 of *Lecture Notes in Computer Science*, pages 283–302. Springer, 2012.

A Example figure of a convex hull of a set of possible transitions

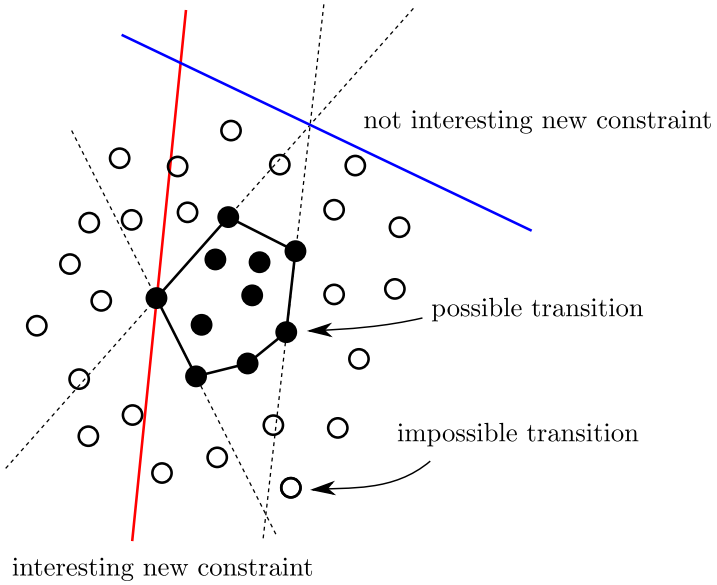

Figure 1: Example of the convex hull of a set of possible transitions. Since in practice transitions only lie on the hypercube $\{0,1\}^m$, this figure is just a sketch to give intuition about our method.

B DDT of PRESENT

Table 10: DDT of the 4-bit Sbox used in PRESENT

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	4	0	0	0	4	0	4	0	0	0	4	0	0
2	0	0	0	2	0	4	2	0	0	0	2	0	2	2	2	0
3	0	2	0	2	2	0	4	2	0	0	2	2	0	0	0	0
4	0	0	0	0	0	4	2	2	0	2	2	0	2	0	2	0
5	0	2	0	0	2	0	0	0	0	2	2	2	4	2	0	0
6	0	0	2	0	0	0	2	0	2	0	0	4	2	0	0	4
7	0	4	2	0	0	0	2	0	2	0	0	0	2	0	0	4
8	0	0	0	2	0	0	0	2	0	2	0	4	0	2	0	4
9	0	0	2	0	4	0	2	0	2	0	0	0	2	0	4	0
10	0	0	2	2	0	4	0	0	2	0	2	0	0	2	2	0
11	0	2	0	0	2	0	0	0	4	2	2	2	0	2	0	0
12	0	0	2	0	0	4	0	2	2	2	2	0	0	0	2	0
13	0	2	4	2	2	0	0	2	0	0	2	2	0	0	0	0
14	0	0	2	2	0	0	2	2	2	2	0	0	2	2	0	0
15	0	4	0	0	4	0	0	0	0	0	0	0	0	0	4	4

C Proof and example for merging three balls

C.1 Proof of Proposition 4

We recall that $a, b, c, a \oplus b \oplus c \in \mathcal{I}, b \neq c, \mathrm{wt}(a \oplus b) = \mathrm{wt}(a \oplus c) = 1$,

$$P_a = \mathcal{B}(1, a) \setminus \mathcal{I} \subseteq \mathcal{S}(1, a), \qquad P_b = \mathcal{B}(1, b) \setminus \mathcal{I} \subseteq \mathcal{S}(1, b),$$
$$P_c = \mathcal{B}(1, c) \setminus \mathcal{I} \subseteq \mathcal{S}(1, c), \qquad Q_1 = P_a \oplus a \oplus b \subseteq \mathcal{S}(1, b),$$
$$Q_2 = P_a \oplus a \oplus c \subseteq \mathcal{S}(1, c), \qquad Q_3 = P_b \oplus b \oplus c \subseteq \mathcal{S}(1, c),$$

$A(x) \geq 2, B(x) \geq 2$ and $C(x) \geq 2$ remove respectively $\mathcal{B}(1, a) \setminus (\mathcal{Q} \cup \{c\}), \mathcal{B}(1, b) \setminus \mathcal{Q}$ and $\mathcal{B}(1, c) \setminus \mathcal{Q}$ and we introduce the notations

$$A(x) = \sum_{i=0}^{m-1} \alpha_i (x_i \oplus a_i), \qquad B(x) = \sum_{i=0}^{m-1} \beta_i (x_i \oplus b_i),$$
$$C(x) = \sum_{i=0}^{m-1} \gamma_i (x_i \oplus c_i).$$

We also denote the basis vectors as

$$e_i = (0, \cdots, \underset{\substack{\uparrow \\ i}}{1}, \cdots, 0)$$

for all $i \in [0, m-1]$. Finally, j_b and j_c will denote the indices such that $a \oplus b = e_{j_b}$ and $a \oplus c = e_{j_c}$.

We then have the two following lemmas:

Lemma 1. *The points a, b, c and $a \oplus b \oplus c$ do not belong to $\mathcal{Q}$.*

Proof. First, since those points are impossible transitions, they cannot belong to $P_a \cup P_b \cup P_c$ by definition. Then if $a \in Q_1$, then $b \in P_a$ which as said above is impossible. For all the other cases we have: $a \in Q_2 \Rightarrow c \in P_a$, $a \in Q_3 \Rightarrow a \oplus b \oplus c \in P_b$, $b \in Q_2 \Rightarrow a \oplus b \oplus c \in P_a$, $b \in Q_3 \Rightarrow c \in P_b$, $c \in Q_1 \Rightarrow a \oplus b \oplus c \in P_a$. □

Lemma 2. *It holds that* $\alpha_{j_c} = 2$ *and* $\alpha_{j_b} = \beta_{j_b} = \beta_{j_c} = \gamma_{j_b} = \gamma_{j_c} = 1$.

Proof. From Proposition 3 in the case $d = 1$, we have for all $i \in [0, m-1]$:

- $\alpha_i = 1 + ((c_i \oplus a_i) \vee \bigvee_{q \in \mathcal{Q} \cap \mathcal{B}(1,a)} q_i \oplus a_i)$ because $A(x) \geq 2$ removes $\mathcal{B}(1,a) \setminus (\mathcal{Q} \cup \{c\})$,
- $\beta_i = 1 + \bigvee_{q \in \mathcal{Q} \cap \mathcal{B}(1,b)} q_i \oplus b_i$,
- $\gamma_i = 1 + \bigvee_{q \in \mathcal{Q} \cap \mathcal{B}(1,c)} q_i \oplus c_i$.

Since by definition, $a \oplus c = e_{j_c}$, $\alpha_{j_c} = 2$. If $\alpha_{j_b} = 2$, $\exists q \in \mathcal{Q} \cap \mathcal{B}(1,a) : q_{j_b} \neq a_{j_b}$, i.e. $q = a \oplus e_{j_b} = b$ but $b \notin \mathcal{Q}$. Hence $\alpha_{j_b} = 1$. In the same way, $\beta_{j_b} = 2 \Leftrightarrow b \oplus e_{j_b} = a \in \mathcal{Q}$, $\beta_{j_c} = 2 \Leftrightarrow b \oplus e_{j_c} = a \oplus b \oplus c \in \mathcal{Q}$, $\gamma_{j_b} = 2 \Leftrightarrow c \oplus e_{j_b} = a \oplus b \oplus c \in \mathcal{Q}$ and $\gamma_{j_c} = 2 \Leftrightarrow c \oplus e_{j_c} = a \in \mathcal{Q}$. □

We are ready now to give a proof of Proposition 4.

Proof. Let $x \notin \mathcal{B}(1,a) \cup \mathcal{B}(1,b) \cup \mathcal{B}(1,c)$.

- Let $x_{j_b} = a_{j_b}$ and $x_{j_c} = a_{j_c}$. Then $x \notin \mathcal{B}(1,a) \Rightarrow \exists i_1, i_2 \notin \{j_b, j_c\} : x_{i_1} \neq a_{i_1}$ and $x_{i_2} \neq a_{i_2}$. Hence $A(x) \geq \alpha_{i_1} + \alpha_{i_2} \geq 2$, $B(x) \geq \beta_{i_1} + \beta_{i_2} + \beta_{j_b} \geq 3$ and $C(x) \geq \gamma_{i_1} + \gamma_{i_2} + \gamma_{j_c} \geq 3$.
- Let $x_{j_b} = a_{j_b}$ and $x_{j_c} \neq a_{j_c}$. Then $x_{j_b} = c_{j_b}$ and $x_{j_c} = c_{j_c}$ and $x \notin \mathcal{B}(1,c) \Rightarrow \exists i_1, i_2 \notin \{j_b, j_c\} : x_{i_1} \neq a_{i_1}$ and $x_{i_2} \neq a_{i_2}$. Hence $A(x) \geq \alpha_{i_1} + \alpha_{i_2} + \alpha_{j_c} \geq 4$, $B(x) \geq \beta_{i_1} + \beta_{i_2} + \beta_{j_b} + \beta_{j_c} \geq 4$ and $C(x) \geq \gamma_{i_1} + \gamma_{i_2} \geq 2$.
- Let $x_{j_b} \neq a_{j_b}$ and $x_{j_c} = a_{j_c}$. Then $x_{j_b} = b_{j_b}$ and $x_{j_c} = b_{j_c}$ and $x \notin \mathcal{B}(1,b) \Rightarrow \exists i_1, i_2 \notin \{j_b, j_c\} : x_{i_1} \neq a_{i_1}$ and $x_{i_2} \neq a_{i_2}$. Hence $A(x) \geq \alpha_{i_1} + \alpha_{i_2} + \alpha_{j_b} \geq 3$, $B(x) \geq \beta_{i_1} + \beta_{i_2} \geq 2$ and $C(x) \geq \gamma_{i_1} + \gamma_{i_2} + \gamma_{j_b} + \gamma_{j_c} \geq 4$.
- Let $x_{j_b} \neq a_{j_b}$ and $x_{j_c} \neq a_{j_c}$. Then $x \notin \mathcal{B}(1,b) \cup \mathcal{B}(1,c) \Rightarrow \exists i_b, i_c \notin \{j_b, j_c\} : x_{i_b} \neq a_{i_1}$ and $x_{i_c} \neq a_{i_2}$.
  - If $i_b \neq i_c$, $A(x) \geq 4$, $B(x) \geq 3$ and $C(x) \geq 3$.
  - Otherwise, $A(x) = \alpha_{i_b} + \alpha_{i_c} + \alpha_{j_c} \geq 4$, $B(x) = \beta_{i_b} + \beta_{j_c} \geq 2$ and $C(x) = \gamma_{i_b} + \gamma_{j_b} \geq 2$.

Therefore, in all the above cases the inequality $A(x) + B(x) + C(x) \geq 8$ is verified.

Let now $x \in (\mathcal{B}(1,a) \cup \mathcal{B}(1,b) \cup \mathcal{B}(1,c)) \cap \mathcal{Q}$.

- If $x \in \mathcal{B}(1,a)$, since $a, b, c \notin \mathcal{Q}$, $\exists i \notin \{j_b, j_c\} : x = a \oplus e_i$. It immediatly follows that $\alpha_i = 2$. Moreover, $b \oplus e_i = x \oplus a \oplus b \in \mathcal{B}(1,b) \cap \mathcal{Q}$ so $\beta_i = 2$ and $c \oplus e_i = x \oplus a \oplus c \in \mathcal{B}(1,c) \cap \mathcal{Q}$ and thus $\gamma_i = 2$. Hence $A(x) = \alpha_i = 2$, $B(x) = \beta_i + \beta_{j_b} = 3$ and $C(x) = \gamma_i + \gamma_{j_c} = 3$. Finally, $A(x) + B(x) + C(x) = 8$.
- If $x \in \mathcal{B}(1,b)$, since $a, b, a \oplus b \oplus c \notin \mathcal{Q}$, $\exists i \notin \{j_b, j_c\} : x = b \oplus e_i$. We have $A(x) = \alpha_i + \alpha_{j_b}$, $B(x) = \beta_i = 2$ and $C(x) = \gamma_i + \gamma_{j_b} + \gamma_{j_c}$. Since $x = a \oplus e_i \oplus a_{j_b} = c \oplus e_i \oplus e_{j_b} \oplus e_{j_c}$, $x \in \mathcal{Q}$ but $x \notin \mathcal{S}(1,a) \cup \mathcal{S}(1,c)$. Hence $x \in P_b$ or $x \in Q_1$.
  - If $x \in P_b$, $c \oplus e_i = x \oplus b \oplus c \in Q_3 \subseteq \mathcal{Q} \cap \mathcal{B}(1,c)$ so $\gamma_i = 2$.
  - If $x \in Q_1$, $a \oplus e_i = x \oplus a \oplus b \in P_a \subseteq \mathcal{Q} \cap \mathcal{B}(1,a)$ so $\alpha_i = 2$.

In both cases, $A(x) + B(x) + C(x) \geq 8$.

- If $x \in \mathcal{B}(1, c)$, since $a, c, a \oplus b \oplus c \notin \mathcal{Q}$, $\exists i \notin \{j_b, j_c\} : x = c \oplus e_i$. We have $A(x) = \alpha_i + \alpha_{j_c} = \alpha_i + 2$, $B(x) = \beta_i + \beta_{j_b} + \beta_{j_c}$ and $C(x) = \gamma_i = 2$. Hence $A(x) + B(x) + C(x) \geq 8$.

Finally, let $x \in (\mathcal{B}(1, a) \cup \mathcal{B}(1, b) \cup \mathcal{B}(1, c)) \setminus \mathcal{Q}$.

- If $x \in \{a, b, c, a \oplus b \oplus c\}$, one can easily check that $A(x) + B(x) + C(x) \leq 7$.
- If $x \in \mathcal{B}(1, a) \setminus \{a, b, c\}$, $\exists i \notin \{j_b, j_c\} : x = a \oplus e_i$. $A(x) = \alpha_i$, $B(x) = \beta_i + \beta_{j_b} = \beta_i + 1$ and $C(x) = \gamma_i + \gamma_{j_c} = \gamma_i + 1$. Since $x = a \oplus e_i \notin \mathcal{Q}$, we have $A(x) = \alpha_i = 1$.
- If $x \in \mathcal{B}(1, b) \setminus \{a, b, a \oplus b \oplus c\}$, $\exists i \notin \{j_b, j_c\} : x = b \oplus e_i$. $A(x) = \alpha_i + \alpha_{j_b} = \alpha_i + 1$, $B(x) = \beta_i = 1$ and $C(x) = \gamma_i + \gamma_{j_b} + \gamma_{j_c} = \gamma_i + 2$. If $\alpha_i = 2$, $a \oplus e_i \in \mathcal{Q}$ then $a \oplus e_i \in P_a$ and $x = (a \oplus e_i) \oplus a \oplus b \in Q_1 \subseteq \mathcal{Q}$. Hence $\alpha_i = 1$.
- If $x \in \mathcal{B}(1, c) \setminus \{a, c, a \oplus b \oplus c\}$, $\exists i \notin \{j_b, j_c\} : x = b \oplus e_i$. $A(x) = \alpha_i + \alpha_{j_c} = \alpha_i + 2$, $B(x) = \beta_i + \beta_{j_b} + \beta_{j_c} = \beta_i + 2$ and $C(x) = \gamma_i = 1$. If $\alpha_i = 2$, $a \oplus e_i \in \mathcal{Q}$ then $a \oplus e_i \in P_a$ and $x = (a \oplus e_i) \oplus a \oplus c \in Q_2 \subseteq \mathcal{Q}$. Hence $\alpha_i = 1$. If $\beta_i = 2$ then $b \oplus e_i \in \mathcal{Q}$.
  - If $b \oplus e_i \in P_b$, $x \in Q_3 \subseteq \mathcal{Q}$.
  - If $b \oplus e_i \in Q_1$, $a \oplus e_i \in P_a$ and $x = (a \oplus e_i) \oplus a \oplus c \in Q_2 \subseteq \mathcal{Q}$.

  Hence $\beta_i = 1$.

In conclusion, we always have in this last case that $A(x) + B(x) + C(x) \leq 7$.

□

C.2 Example of Algorithm 3 on PRESENT

Example 5. Let $a = [0, 11]$, $b = [0, 15]$ and $c = [0, 10]$ be three impossible transition points for the Sbox of PRESENT.

$$\mathcal{B}(1, [0, 11]) = \{[0, 11], [0, 10], [0, 9], [0, 15], [0, 3], [1, 11], [2, 11], [4, 11], [8, 11]\},$$
$$\mathcal{B}(1, [0, 15]) = \{[0, 15], [0, 14], [0, 13], [0, 11], [0, 7], [1, 15], [2, 15], [4, 15], [8, 15]\}$$
$$\mathcal{B}(1, [0, 10]) = \{[0, 10], [0, 11], [0, 8], [0, 14], [0, 2], [1, 10], [2, 10], [4, 10], [8, 10]\}.$$

Here, $P_a = \{[8, 11]\}$, $P_b = \{[8, 15]\}$ and $P_c = \{[2, 10], [4, 10]\}$. These sets correspond to all possible transitions inside each ball and therefore they should not be removed by the final inequality. We note them in red.

Then we also compute the three sets Q_1, Q_2 and Q_3, each one containing possible transitions but also impossible transition points that will unfortunatly not be discarded by the new inequality. We highlight these points in blue unless they are already in red. By following notation we get that $Q_1 = \{[8, 15]\}$, $Q_2 = Q_3 = \{[8, 10]\}$. With the set $\mathcal{Q}$, we construct the following distorted balls

$$\mathcal{B}(1, [0, 11]) \setminus (\mathcal{Q} \cup \{[0, 10]\}), \quad \mathcal{B}(1, [0, 15]) \setminus \mathcal{Q} \text{ and } \mathcal{B}(1, [0, 10]) \setminus \mathcal{Q}.$$

Following the technique of Proposition 3 for each of the three distorted balls, we get the following three linear constraints:

$$x_0 + x_1 + x_2 + 2x_3 + 2(1 - y_0) + (1 - y_1) + y_2 + (1 - y_3) \geq 2$$
$$x_0 + x_1 + x_2 + 2x_3 + (1 - y_0) + (1 - y_1) + (1 - y_2) + (1 - y_3) \geq 2$$
$$x_0 + 2x_1 + 2x_2 + 2x_3 + y_0 + (1 - y_1) + y_2 + (1 - y_3) \geq 2$$

Directly adding the three inequalities, while mathematically correct, usually yields inequalities that remove a smaller subset of impossible transitions than what we could get. For this reason, we use the following subtlety: we add 2 to the right-hand side of the sum to construct an inequality removing only the black points inside each ball. By doing so we get that

$$3x_0 + 4x_1 + 4x_2 + 6x_3 + 2(1 - y_0) + 3(1 - y_1) + y_2 + 3(1 - y_3) \geq 6$$

removes the 17 points

$$\begin{aligned}&\{[0,2],[0,3],[0,7],[0,8],[0,9],[0,10],[0,11],[0,13],[0,14]\\ &\ [0,15],[1,10],[1,11],[1,15],[2,11],[2,15],[4,11],[4,15]\} =\\ &(\mathcal{B}(1,a) \cup \mathcal{B}(1,b) \cup \mathcal{B}(1,c)) \setminus \{[2,10],[4,10],[8,10],[8,11],[8,15]\}.\end{aligned}$$

D On the complexity of algorithms for linear layer modeling

For our implementation of Algorithms 4 and 5, we represented binary matrices as vectors of 64-bit integers, each integer representing a row. The multiplication of an n-bit binary vector with a $n \times N$ matrix then takes at most n `XOR`s of 64-bit integers when $N \leq 64$. With this implementation, the running time of Algorithm 4 on the AES `MixColumns` matrix on a single core is 929 seconds. In Algorithm 5, the computation of Q applies Algorithm 4 on $2b$ matrices with $\frac{n}{b}$ rows. The complexity is then

$$2b \cdot \left(\frac{n}{b}\right)^2 \cdot 2^{\frac{n}{b}-1} \texttt{ XOR}\text{s of 64-bit integers.}$$

The running time of this computation is then negligible compared to the computation of P in practice. Indeed, the computation of P applies Algorithm 4 on a matrix with n rows, which has a complexity of

$$n^2 \cdot 2^{n-1} \texttt{ XOR}\text{s of 64-bit integers.}$$

Finally, the running time of Algorithm 5 is $(p + 1)$ times the running time of Algorithm 4.

E `MixColumns` modeling for the AES

Algorithm 5 applied on the AES `MixColumns` outputted the matrices of Table 11. The lines are represented as hexadecimal numbers with the first column element being the least significant bit (i.e. the rightmost one).

F Analyzed linear layers

We provide here a brief description of the linear layers of the ciphers Anubis, Aria and Saturnin whose modeling for MILP we analyzed in Section 3.

- Anubis is a 128-bit block cipher designed by Barreto and Rijmen in 2000 in the context of the NESSIE project. Anubis follows the SPN construction and uses an involutive `MixColumns` operation, based on an MDS matrix

$$H = \begin{pmatrix} 1 & 2 & 4 & 6 \\ 2 & 1 & 6 & 4 \\ 4 & 6 & 1 & 2 \\ 6 & 1 & 2 & 4 \end{pmatrix} \in \mathcal{M}_4(\mathbb{F}_{2^8}).$$

Table 11: AES `MixColumns` matrices

A	P	Q_0	PAQ
101018180	10101	81	1010101000080
202028381	2	2	202020301
404040602	4	4	404040602
808088c84	8000808	88	8000808840c0000
1010109888	10	10	1010101808
2020203010	20	20	2020203010
4040406020	40	40	4040406020
808080c040	80	80	808080c040
10001818001	1010001		101000100800100
20002838102	200		20002030102
40004060204	400		40004060204
800088c8408	8080800		80808000000840c
100010988810	1000		100010180810
200020301020	2000		200020301020
400040602040	4000		400040602040
800080c04080	8000		800080c04080
1000081800101	1010100		101010000008001
2000083810202	20000		2000003010202
4000006020404	40000		4000006020404
800008c840808	80808		808080c000084
10000098881010	100000		10000018081010
20000030102020	200000		20000030102020
40000060204040	400000		40000060204040
800000c0408080	800000		800000c0408080
100000080010181	1000101		100010180010000
200000081020283	2000000		200000001020203
400000002040406	4000000		400000002040406
80000008408088c	8080008		808000800840c00
1000000088101098	10000000		1000000008101018
2000000010202030	20000000		2000000010202030
4000000020404060	40000000		4000000020404060
80000000408080c0	80000000		80000000408080c0

$$Q = \begin{pmatrix} Q_0 & & & & & & & \\ & Q_0 & & & & & & \\ & & Q_0 & & & & & \\ & & & Q_0 & & & & \\ & & & & I & & & \\ & & & & & I & & \\ & & & & & & I & \\ & & & & & & & I \end{pmatrix}$$

Each entry of H is an element $\sum x_i X^i$ of the finite field $\mathbb{F}_2[X]/(X^8+X^4+X^3+X^2+1)$ represented by the integer $\sum x_i 2^i$. For the Anubis's `MixColumns` operation, the quantity of Equation (4) initially at 7168 drops to 2032 after applying Algorithm 4 and to 1680 after applying Algorithm 5.

- Saturnin is a 2-round candidate of the NIST Lightweight Crypto Competition,

designed by Canteaut et al. [CDL+19]. Its main component is a block cipher with a `MixColumns` operation applying the transformation $M : (\mathbb{F}_2^4)^4 \leftarrow (\mathbb{F}_2^4)^4$ (see Figure 2) to different parts of the state. Algorithm 4 allows the quantity of Equation (4) initially at 5632 to drop down to 352 but Algorithm 5 does not permit to reduce this quantity further.

$$\alpha(x) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \cdot x$$

$$M : \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \mapsto \begin{pmatrix} \alpha^2(a) \oplus \alpha^2(b) \oplus \alpha(b) \oplus c \oplus d \\ a \oplus \alpha(b) \oplus b \oplus \alpha^2(c) \oplus c \oplus \alpha^2(d) \oplus \alpha(d) \oplus d \\ a \oplus b \oplus \alpha^2(c) \oplus \alpha^2(d) \oplus \alpha(d) \\ \alpha^2(a) \oplus a \oplus \alpha^2(b) \oplus \alpha(b) \oplus b \oplus c \oplus \alpha(d) \oplus d \end{pmatrix}$$

Figure 2: Saturnin's M transformation.

- Aria. Presented at ICISC in 2003 by Kwon et al. [KKP+03], Aria is an SPN block cipher whose diffusion layer consists in applying a 16×16 binary matrix (given in Figure 3) to different parts of the state. The quantity of Equation (4) is 2048 and is not improved by our algorithms.

$$\begin{pmatrix}
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Figure 3: Aria diffusion layer matrix.

G More statistics on the experiment of Section 4

Table 12: Computation time (sec) for 32 rounds of SKINNY-128 on a sample of 1975 input/output pairs.

`Sbox`	Alg. 2		Alg. 2 and 3	
`MixColumns`	Naïve	Improved	Naïve	Improved
Mean	185	**21**	172	29
Std	359	26	364	34
Min	5	5	5	5
25 %	6	6	6	6
50 %	61	**6**	7	**6**
75 %	106	**7**	104	63
Max	1876	**120**	2084	292

Table 13: Computation time (sec) for 5 rounds of AES on a sample of 4594 dense input/output pairs.

`Sbox`	Alg. 2		Alg. 2 and 3	
`MixColumns`	Naïve	Improved	Naïve	Improved
Mean	43	40	**16**	25
Std	2	7	4	28
Min	22	17	**8**	**8**
25 %	40	40	**11**	19
50 %	42	41	**13**	20
75 %	44	44	**20**	22
Max	55	53	**52**	379

Table 14: Computation time (sec) for 5 rounds of AES on a sample of 259 sparse input/output pairs.

`Sbox`	Alg. 2		Alg. 2 and 3	
`MixColumns`	Naïve	Improved	Naïve	Improved
Mean	1599	1944	**441**	662
Std	1560	2051	166	274
Min	17	24	**7**	11
25 %	983	1047	**382**	460
50 %	1194	1465	**424**	702
75 %	2036	2347	**488**	822
Max	18237	23566	**1062**	2824

One can see in those experiments that the computation can be very long and that the standard deviation is huge. We do not have any explanation for this fact. However, the means and quantiles behave coherently with each other, which shows that a different MILP modeling does have a clear influence on the solving time.