Minimalist Model for Impossible Differentials

Patrick Derbez[1] and Marie Euler[1,2]

[1] Univ Rennes, Inria, Centre National de la Recherche Scientifique (CNRS), Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), Rennes, France
patrick.derbez,marie.euler@irisa.fr
[2] DGA MI, Bruz, France

Keywords: Impossible differential · MILP · AES · ARADI · MIDORI · PRESENT · SIMON · Simeck · SKINNY · SPECK

Abstract. This paper introduces a new MILP modeling to find impossible differential (ID) distinguishers and attacks. Standard models for ID are negative models, in the sense that a differential is impossible if and only if the model has no solution. Our new modelling technique focuses on probable ID, differentials that are probably impossible. While this might lead to false positives, the main advantage is that searching for such probable ID can be achieved through a positive model. This facilitates the search for the best impossible differential attacks without first exhausting all possible ID distinguishers on a target. We also propose to simplify the modelling by only considering two possible states for internal cells: inactive and unknown. In this case there are no longer direct contradictions but only indirect ones, assuming that it is impossible that all cells are inactive.
With these two simple ideas, we are able to retrieve the longest impossible differentials distinguishers on `MIDORI`, `SKINNY`, `PRESENT`, `SIMON`, `Simeck` and `SPECK`. Furthermore, as the model looking for candidates is based on satisfiability, it can be incorporated in a larger model which looks directly for the best attacks in order to enumerate the distinguishers in the order of the complexity of the associated attacks, which we did for the `AES`, `ARADI`, `SIMON` and `SKINNY`.

1 Introduction

Impossible differential (ID) cryptanalysis is a powerful method for eliminating incorrect keys by exploiting the observation of an impossible event. In this technique, the attacker partially encrypts and decrypts a pair of messages through a cipher while guessing round-key bits to uncover an impossible differential transition in the middle rounds. Specifically, this transition involves a pair of differences (Δ_X, Δ_Y) that occurs with zero probability in the remaining rounds of the cipher. Such a pair of differences is referred to as an *impossible differential distinguisher*. If this event is observed, the corresponding key guess is discarded as invalid. The attack concludes with an exhaustive search over the remaining key candidates.

This technique was introduced simultaneously in 1998 by Knudsen [22], who applied it to his `AES` candidate `DEAL`, and by Biham et al. [7], who successfully attacked 31 out of 32 rounds of `SkipJack`. Since its inception, ID cryptanalysis has proven to be a robust cryptanalytic tool. For instance, it remains one of the most effective techniques against 7 rounds of `AES-128` [9].

The complexity of an ID attack is largely determined by the *quality* of the impossible differential distinguisher. As such, much of the prior research has focused on techniques to identify long impossible distinguishers. The primary method relies on the *miss-in-the-middle* approach introduced by Biham et al. [7]. In this approach, the forward propagation of Δ_{in} with probability one contradicts the backward propagation of Δ_{out}, also with probability one, after a few rounds. Initially applied manually, this concept has been extended through a variety of automated techniques, including the $\mathcal{U}$-method [21], the UID-method [26], and the WW-method [40]. These methods rely on the propagation of four types of differences: zero difference, fixed nonzero difference, unspecified nonzero difference, and unknown difference.

In 2017, Sasaki and Todo [33] introduced a new approach leveraging Mixed-Integer Linear Programming (MILP). Their method involves modeling the difference propagation while fixing input and output activity patterns. If the MILP solver cannot find a solution (i.e., a valid differential characteristic), the differential is deemed impossible. However, for S-boxes operating on more than 4 bits, the MILP model often becomes too slow to yield results. To address this, they proposed an approximation using arbitrary S-boxes, where any active input bit implies a fully active output. Although this approximation simplifies the search, it sacrifices precision, as it does not account for detailed compatibility between successive S-boxes. This technique allows analysts to identify impossible differentials without requiring manual anticipation of contradiction patterns. However, it has drawbacks, such as a lack of explainability for why a differential is impossible and the need to test multiple input and output activity patterns separately, which becomes cumbersome for exhaustive security proofs.

To mitigate the issue of exhaustive testing, Sun et al. [35] demonstrated in 2016 that in the arbitrary S-box model, for SPNs or Feistel ciphers with SP-type round functions, any impossible differential of r rounds implies the existence of a weight-1 ID distinguisher for the same number of rounds. This result reduces the number of required activity pattern tests from 2^{2m} to m^2, where m is the number of bits or cells in the input.

In 2022, Hu, Peyrin, and Wang [19] introduced a new method for proving the security of specific ciphers without relying on the arbitrary S-box approximation. Their approach partitions the space of ID candidates and uses an MILP model incorporating the S-box difference distribution table (DDT). This advancement improves accuracy in identifying impossible differentials but retains the limitations of being a *negative model*, where a distinguisher's impossibility is proven only if no solutions exist.

Both MILP-based approaches share a key drawback: they are negative models that cannot be directly integrated into broader frameworks for finding the most efficient attacks. Furthermore, the low-weight ID distinguishers suggested by these models may not always lead to the most effective attacks.

Since 2023, Hadipour et al. [11, 17, 18] have introduced a Constraint Programming (CP)-based adaptation of the $\mathcal{U}$-method. Initially limited to strongly aligned ciphers with predefined contradiction locations [18], their models have evolved to handle bitwise representations, incorporate advanced key-recovery strategies [17], and even address indirect contradictions in ARX ciphers [11]. This line of research offers two major advantages: positive modeling, which facilitates integration into broader attack optimization frameworks, and flexibility in application across various cipher classes. Despite these advancements, Hadipour et al. have primarily focused on identifying *good* distinguishers before optimizing the corresponding key-recovery steps.

The first tool explicitly designed to find optimal ID attacks is that of Derbez and Fouque [14], which does not rely on generic solvers. However, their algorithm remains somewhat opaque, as the paper lacks detailed information on its inner workings, making it difficult to compare with other approaches. Recently, the automatic cryptanalysis tool CLAASP has been improved using the recent works of Hadipour *et al.* to generate CP models searching for attacks. It has been applied to both `LBlock` and `HIGHT` [5] but does not include the indirect contradiction search.

Our contributions In this work, we address the challenge of designing a simpler positive MILP model that can seamlessly handle cell-based ciphers, bit-oriented ciphers, AndRX and ARX ciphers. Our approach eliminates the need for analysts to exhaustively enumerate all contradictions manually. Furthermore, we demonstrate how this model can be integrated into a comprehensive framework for directly searching for the best impossible differential attacks in a unified process.

The core idea is to combine the validation strategy of Sasaki and Todo [33], which checks the validity of impossible differentials, with a novel positive model. This new model generates activity patterns that are likely to lead to valid impossible differential distinguishers. Furthermore, when included in full model searching directly for attacks, only few *probable* ID distinguishers are tried for. As a second contribution, we also propose to simplify the propagation of differences. Unlike the $\mathcal{U}$-method, which propagates four types of information (zero difference, fixed nonzero difference, unspecified nonzero difference, and unknown difference), our model simplifies this propagation to just two types: zero difference and unknown difference. This simplification reduces complexity while maintaining effectiveness.

The source code of our experiments is available at:

`https://gitlab.inria.fr/capsule/idtool.git`

Organisation The rest of the paper is organized as follows. In Section 2, we review in more details impossible differential attacks, and in particular their complexity. In Section 3, we explain in detail how our modeling works. Section 4 is dedicated to applications of the modeling to different contexts. Two full integrations in attack frameworks will be demonstrated with the cases of `AES` and `SKINNY-128` (and in appendix the low-latency design `ARADI`). More details on the distinguisher modeling will be given taking into example the word-oriented cipher `Midori`, the AndRX ciphers `SIMON` and the AddRX cipher `SPECK`. Distinguishers for the bit-oriented cipher `PRESENT` and the AndRX cipher `Simeck` are also presented in appendix.

2 Principle of ID attacks

2.1 Impossible Differential attacks

We begin by recalling the framework for impossible differential attacks on block ciphers, as outlined in [10]. Consider an n-bit block cipher with a k-bit master key K. The attack leverages a zero-probability differential distinguisher spanning r_d rounds, which begins with an n-bit input difference (or set of differences) Δ_X and concludes with an n-bit output difference (or set of differences) Δ_Y, as illustrated in Figure 1. To extend the attack, r_{in} rounds are prepended to the distinguisher and r_{out} rounds are added afterward.

$$\begin{tikzcd}
\Delta_{\text{in}} \arrow[r, "r_{\text{in}}"] & \Delta_X \arrow[r, leftrightarrow, "r_{\text{D}}", "\times" marking] & \Delta_Y & \Delta_{\text{out}} \arrow[l, "r_{\text{out}}"']
\end{tikzcd}$$

Fig. 1: The impossible differential attack principle : a central distinguisher surrounded by key recovery rounds

The set of potential differences at the plaintext side, denoted Δ_{in}, consists of differences that may propagate to Δ_X after r_{in} rounds. Similarly, Δ_{out} represents the set of differences at the ciphertext side that may result from Δ_Y after r_{out} rounds. The probability that a difference in Δ_{in} propagates to Δ_X is denoted as $2^{-c_{\text{in}}}$, while $2^{-c_{\text{out}}}$ represents the probability of a difference in Δ_{out} propagating to Δ_Y. Finally, the set of key bits involved in determining whether a difference in Δ_{in} leads to Δ_X is k_{in}, while k_{out} denotes the corresponding set of key bits at the ciphertext side.

Given a pair of messages with a plaintext difference in Δ_{in} and a ciphertext difference in Δ_{out}, the probability that a specific key guess results in Δ_X and Δ_Y (and is thus discarded) is $2^{-c_{\text{in}}-c_{\text{out}}}$. Consequently, the probability of not discarding a given key is $(1 - 2^{-c_{\text{in}}-c_{\text{out}}})$. When considering N such pairs, the probability of not discarding a given key becomes approximately $(1 - 2^{-c_{\text{in}}-c_{\text{out}}})^N \approx \exp(-N \cdot 2^{-c_{\text{in}}-c_{\text{out}}})$. Furthermore, to construct N pairs, the attacker organizes encryption queries into structures, either on the plaintext side or the ciphertext side. Depending on the required number of pairs, multiple structures may need to be encrypted, or in some cases, fewer than a full structure. Combining these scenarios, the data complexity D is given by the following formula also known as the *Limited Birthday formula* [10]:

$$D = \max\left\{ \min_{\varDelta \in \{\varDelta_{\text{in}}, \varDelta_{\text{out}}\}} \left\{ \sqrt{N \cdot 2^{n+1-|\varDelta|}} \right\}, N \cdot 2^{n+1-|\varDelta_{\text{in}}|-|\varDelta_{\text{out}}|} \right\},$$

where $2^{|\varDelta_{\text{in}}|}$ and $2^{|\varDelta_{\text{out}}|}$ represent the number of differences in $\varDelta_{\text{in}}$ and $\varDelta_{\text{out}}$, respectively. In the classical scenario the data complexity must satisfy $D \leq 2^n$ (the full codebook size) but for tweakable block ciphers it might be extended by the size of the tweak.

The complexity of an impossible differential attack is then given by two main components: the generation of the pairs and the attack itself. As stated in [10] and confirmed in [13], a lower bound for the time complexity T is:

$$T = \left(D + \left(N + \frac{2^{|k_{\text{in}} \cup k_{\text{out}}|} \cdot N}{2^{c_{\text{in}} + c_{\text{out}}}} \right) C'_E \right) C_E,$$

where C_E represents the cost of a single encryption and C'_E the ratio of the cost of a partial encryption to that of a full encryption. This time complexity must satisfy $T < 2^k C_E$, otherwise the attack would not be faster than a brute force. Finally, the memory complexity corresponds to storing N pairs.

Note that the attacker's goal is to reduce the set of possible keys by at least a factor of 2 to ensure that the final exhaustive search step remains computationally feasible or to guess an extra bit and run another attack. We denote by N_{min} the minimum number of pairs required to achieve this reduction. We have:

$$(1 - 2^{-c_{\text{in}} - c_{\text{out}}})^{N_{\text{min}}} < \frac{1}{2}.$$

Using a well-known approximation, this can be simplified into $N_{\text{min}} > \ln(2) \cdot 2^{c_{\text{in}} + c_{\text{out}}}$, which can be rewritten as $N_{\text{min}} > 2^{c_{\text{in}} + c_{\text{out}} - 0.53}$.

2.2 Searching for Impossible Differential distinguishers

As explained in the introduction, searching for impossible differential distinguishers typically involves analyzing three trails: a **forward trail**, which propagates the input difference $\varDelta_X$ with probability 1 in the forward direction; a **backward trail**, which propagates the output difference $\varDelta_Y$ in the backward direction; and a **summary trail**, which consolidates the information deduced independently from the forward and backward trails.

In this context, a bit or cell (depending on the granularity) can take one of three possible states: 0 (no difference), 1 (non-zero difference), or $*$ (undetermined state). A *direct contradiction* arises when a cell is assigned conflicting values in the forward and backward trails (e.g., 0 in one trail and 1 in the other) or when a contradiction emerges during the evaluation of a cipher operation. For example, it may be impossible for an S-box to exhibit a transition such as $(1, 0, *, *) \rightarrow (0, *, *, 1)$.

An *indirect contradiction*, on the other hand, occurs when certain $*$ states must be resolved (to 0 or 1) before reaching a direct contradiction. For instance, in the MixColumns operation of AES, if there are four $*$ states and four 0 states, it is well known that the four $*$ states must resolve to 0. These newly determined values may then lead to a direct contradiction. The first paper pointing out the existence of indirect contradictions used the example of the SIMECK block cipher [30]. Initially found manually, these contradictions were later found with a MILP modeling in [32]. Yet, this was a negative model which therefore did not have to deal explicitly with the progressive deductions.

Indeed, the challenge with indirect contradictions lies in the iterative nature of the deductions required to reach a contradiction. For example, given two consecutive S-boxes, S_1 and S_2, it may be necessary to iteratively refine the $*$ states by alternating deductions between S_1 and S_2 multiple times before a contradiction is identified. Notably, the model proposed by Chakraborty *et al.* in [11] does not account for such indirect contradictions, as their approach limits deductions to a single pass from S_1 to S_2, even though it is unclear whether any of the best distinguishers requires multiple passes. More precisely, once the summary trail is computed, they start from one round (given by the user) and then propagate this state with probability one to both the input and output of the distinguisher and check whether a contradiction with either the forward or the backward trail occurs. They did so because in constraints programming new variables have to be used for each new pass, making the model very complex.

Moreover, some contradictions are not local: several deductions have to combined to obtain the source of the contradiction. It is the case of some impossible differentials on 9 rounds of the Feistel network Lilliput described in [33]: for $\alpha \in \{2, 3, 9, e, f\}$, the difference $(000000\alpha0; 00000000)$ cannot lead to the difference $(00000000; 00000\alpha00)$. The contradiction which uses the DDT of Lilliput's S-Box S comes from the fact that for such α there are no β, x, y, z such that

$$\begin{cases} S(x) \oplus S(x \oplus \alpha) = \beta; \\ S(y) \oplus S(y \oplus \alpha \oplus \beta) = \beta; \\ S(z) \oplus S(z \oplus \alpha) = \alpha \oplus \beta. \end{cases}$$

This set of contradictory equations arise from the observations of several Feistel XORs thus could not have been obtained with predefined local variables indicating potential sources of contradictions.

3 New search principle

Previous research has shown that handling all indirect contradictions using a *positive* model—where a solution represents an impossible distinguisher—is highly challenging. In this work, we focus on indirect contradictions, as they are the most difficult to address. Our approach is conceptually simple yet highly effective. Specifically, at the onset of an indirect contradiction, there exist $*$ states that must be resolved into either 0 or 1. This resolution is the core of our method for identifying impossible differential distinguishers and attacks.

Rather than directly searching for impossible differential distinguishers, our model targets distinguishers where, in the summary trail, at least one $*$ variable resolves to either 0 or 1. This approach allows us to identify *probable* impossible differential distinguishers, which can then be further analyzed externally to verify their validity.

To achieve this, we introduce two MILP models. The first, called the **generator model**, identifies probable impossible differential distinguishers. The second, referred to as the **validator model**, verifies the results of the generator model. While other constraint programming languages could be used to implement these models, it is crucial for the generator model to support callback functionality. This allows external programs to interact with the solver during the solving process, significantly enhancing efficiency. Additionally, it is preferable for the generator model to incorporate the full attack directly (not only the distinguisher part) and to optimize on the time complexity, ensuring it focuses only on probable distinguishers that lead to the best attacks. These two models are nested: the main interface is the one of the generator model which is parameterized by the number of rounds of either the distinguisher r_D or the respective parts of the attack $(r_{\text{in}}, r_D, r_{\text{out}})$ depending on whether the user wants to find a distinguisher or an attack. The validator model is called by the generator model to validate the distinguisher candidates as soon as they are generated. It never has to be called directly.

Finally, we emphasize that while the generator model is a positive model, the validator model does not need to be. This dual-model approach is deliberate and permits to simplify the modeling while handling all the contradictions that a *negative* model can find. However, its efficiency highly depends on how many candidates are generated by the generator model.

Simplifying the model. In practice, the propagation of 1's in impossible differential distinguishers is highly ineffective within a cipher. At the cell level, the XOR of two 1's yields an indeterminate result, and at the bit level, it is uncommon for 1's to persist through more than few non-linear operations. Consequently, we propose simplifying the model by removing 1's and focusing solely on 0 and $*$ states. Under this approach, a differential is deemed impossible if and only if all the $*$ states in the summary trail must resolve to 0. In particular, a contradiction is necessary an indirect contradiction. The intuition behind this simplification is that, for the longest impossible differential, the contradiction arises not from the propagation of 1's from both the input and output of the distinguisher, but rather from the propagation of 0's from at least one of these two sides.

Hence, in the following we will describe a simplified model, with only two possible states for each variable – 0 and $*$ – and a distinguisher will be a probable impossible differential if and only if at least one of the $*$ variable of the summary trail has to resolve to 0 to be valid. However, the same approach can obviously be used in a model containing the 1's.

3.1 Generator model: The distinguisher part

Classically, we use three trails in the distinguisher part of the generator model: the forward trail propagates Δ_X with probability 1 in the forward direction, the backward trail propagates Δ_Y in the backward direction and summary trail which merges the information of both these two trails. Each of the variables representing the state values considered in these trails are binary: it is zero if the cell is inactive (denoted 0) and 1 if the difference is unknown (denoted $*$).

We begin by describing the propagation of differences in the forward trail, constraints for the backward trail being similar.

Modeling the XOR and the AND We know that the XOR (resp. the AND) of two inactive cells remains inactive while the result of the XOR (resp. the AND) of any unknown difference with another difference is unknown. Thus we model the XOR $c = a \oplus b$ and the AND $c = a \cdot b$ with the constraints $c \leq a + b \leq 2c$.

Modeling Arbitrary S-Boxes We consider that except if all the inputs of a n-bit S-Box are inactive, the output difference bits of this S-Box are all unknown. Let us denote as $(a_i)_{0 \leq i < n}$ the input differences and $(b_i)_{0 \leq i < n}$ the output differences. The constraints we set on a and b are quite natural:

$$\begin{cases} b_0 = b_1 = \ldots = b_{n-1} \\ \sum a_i / n \leq b_0 \leq \sum a_i \end{cases}$$

Note that, for cell-oriented models, the constraint is even simpler since it simplifies to $a = b$.

Moreover, these kind of modeling naturally captures truncated distinguishers. Indeed, when the S-boxes are deemed arbitrary, the only important information in any bit activity pattern in Δ_X, is the knowledge of which cells are activated. Therefore, when only searching for distinguishers, we can fix an arbitrary bit pattern at the cell-level to indicate that the cell is active, for both the first and last layers of S-boxes. This will limit the number of equivalent solutions of the model. For instance, the constraint $a_0 = a_1 = \cdots = a_{n-1}$ can be applied on the input of the first S-box layer.

Modeling Specific S-Boxes This only applies for bit-oriented models. Given a n-bit S-box, we need to determine for each $(a_0, \ldots, a_{n-1}) \in \{0, *\}^n$ at its input, what is the corresponding output. We first have to compute the list of all possible differential transitions through the S-box and then, for each input, we check which output bits (if any) are set to 0 for all the possible corresponding output differences. Next, we can construct the outputs $(b_0, \ldots, b_{n-1}) \in \{0, *\}^n$ associated to each $(a_0, \ldots, a_{n-1})$ step by step, ordering the inputs by their number of $*$. Finally, for each of the 2^n transitions $(a_0, \ldots, a_{n-1}) \to (b_0, \ldots, b_{n-1})$ we add the constraints:

$$\begin{cases} \sum\limits_{i, b_i = *} (1 - b_i) \leq |\{i, b_i = *\}| \times \sum\limits_{i, a_i = *} (1 - a_i) \\ \sum\limits_{i, b_i = 0} b_i \leq |\{i, b_i = 0\}| \times \sum\limits_{i, a_i = 0} a_i \end{cases}$$

Note that in practice, most of the constraints are redundant and do not need to be included. The final number of constraints only depends on the number of possible $(b_0, \dots, b_{n-1})$. For instance, let assume we have the transitions $(0, *, 0, *) \to (0, 0, *, *)$ and $(*, *, 0, *) \to (0, 0, *, *)$. In this case we can keep only the constraints

$$(1 - b_2) + (1 - b_3) \leq 2((1 - a_1) + (1 - a_3)) \quad \text{and} \quad (1 - b_0) + (1 - b_1) \leq 2a_2,$$

since both $(1-b_2)+(1-b_3) \leq 2((1-a_0)+(1-a_1)+(1-a_3))$ and $(1-b_0)+(1-b_1) \leq 2(a_0 + a_2)$ are weaker. Note that most of the strong S-boxes do not have such patterns and thus the few constraints of the arbitrary S-box modeling are enough. Yet we can observe such weak diffusion pattern for some of the S-boxes used in lightweight ciphers. For example, there are two non-trivial transitions in the 4-bit S-box S of `SKINNY-64`: the third bit of $S(x) \oplus S(x \oplus 1)$ is always zero, which corresponds to the pattern $(*, 0, 0, 0) \to (*, *, 0, *)$, and the fourth bit of $S(x) \oplus S(x \oplus 2))$ is always zero, which is equivalent to the probability one transition $(0, *, 0, 0) \to (*, *, *, 0)$. For all other non null input patterns, the output pattern is merely $(*, *, *, *)$ that is none of the output bits is always inactive. Similarly, the only non-trivial pattern in the 5-bit S-Box of `ASCON` is $(0, 0, 0, 0, *) \to (*, *, 0, *, *)$.

Modeling Linear layers Since we are only looking for propagations of probability 1, the constraints are exactly the same as for XOR: an output bit/cell is 0 if and only if all the bits/cells it depends on the input are 0 as well.

Deduction of the summary trail Let c be the deduction from the knowledge of the cell a in the forward trail and the cell b in the backward trail. The only case where the difference c is unknown is when the differences a and b are unknown. This can be modeled with the constraints $2c \leq a + b \leq 1 + c$.

Modeling the deduction of new zeros This is the novelty of the model regarding the distinguisher. Given an operation f within the cipher, our goal is to identify all subsets of input and output bits or cells such that if all variables in a subset have a difference of 0, then at least one additional input or output bit/cell must also have a difference of 0. For linear layers, this process is straightforward and relies on basic linear algebra to determine which bits or cells can be expressed solely in terms of the subset's variables. For S-boxes, the process is similar to generating the constraints described earlier, but it differs in scale. The number of subsets that must be considered is larger, specifically 2^{n+m} for an n-to-m-bit function f.

We also have to introduce an auxiliary binary variable z into the model, which takes the value 1 for subsets that create a new zero and 0 otherwise. For sake of simplicity and efficiency, we only considered patterns which impose a new zero in a fixed place. While this results in 2^{n+m} inequalities, these can often be reduced using classical techniques such as the Quine-McCluskey algorithm [28] or similar methods.

Let us take as example the 4-bit S-Box of the `PRESENT` blockcipher whose table is [`c`, 5, 6, `b`, 9, 0, `a`, `d`, 3, `e`, `f`, 8, 4, 7, 1, 2]. There are 42 patterns leading to the creation of new zeros. They are presented in Table 1 and actually, all but the last one imply that the S-Box is fully inactive. Note that the $2^{4+4} = 256$ associated constraints are reduced by the Espresso logic minimizer to only 43 constraints.

000*000*	000*00*0	000*00**	000*0*00	000*0*0*	000*0**0
000*0***	000**000	000**0*0	00*0000*	00*000*0	00*00*00
00*0*000	00*0**00	00**00*0	00***000	0*00000*	0*0000*0
0*000*00	0*00*000	0*00*00*	0*0*00*0	0*0**000	0**000*0
0**0*000	0***00*0	0****000	*000000*	*00000*0	*00000**
*0000*00	*0000*0*	*0000**0	*0000***	*000*000	*000*0*0
*00**000	*0*0000*	*0*00*00	**00000*	**000*00	*00**0*0

Table 1: 42 input/output patterns leading to new zeros on the PRESENT S-box. Each pattern is given as the concatenation of the input and the output patterns.

Another interesting example is the specific case of Maximum Distance Separable (MDS) matrices for which the constraints can be significantly simplified. Consider, for instance, the `MixColumns` operation in `AES`, with inputs (a_0, a_1, a_2, a_3) and outputs (b_0, b_1, b_2, b_3). Subsets leading to a new zero must satisfy the following constraints:

$$\begin{cases} a_0 + a_1 + a_2 + a_3 \geq z, \\ b_0 + b_1 + b_2 + b_3 \geq z, \\ a_0 + a_1 + a_2 + a_3 + b_0 + b_1 + b_2 + b_3 \leq 8 - 4z. \end{cases}$$

In essence, this means that no more than three inputs and three outputs can simultaneously be zero; otherwise, new zeros would appear either in the forward or backward trail. However, at least four variables must be zero for new zeros to be created.

Removing invalid candidates If the validator model confirms that a candidate (Δ_X, Δ_Y) is a valid differential, it becomes necessary to exclude this candidate from the solution space of the generator model. Furthermore, we can also eliminate all candidates that include this differential. This is achieved by enforcing a constraint in the model that ensures at least one of the $*$ variables in either Δ_X or Δ_Y is set to 0. This requires adding only a single inequality as a lazy constraint through the callback functionality.

3.2 Generator model : inclusion in an attack framework

The reason we propose a positive model is to include it in a larger model, directly searching for the best impossible differential attack. We describe here the main steps of the key-recovery part.

Active cells The r_{in} rounds of key recovery preceding the distinguisher and the r_{out} rounds following it are modeled in the standard way: Δ_X and Δ_Y are propagated with probability one toward the plaintext and the ciphertext. This process identifies the active cells at the inputs and outputs, as well as the *needed cells* whose values need to be determined to compute the active ones. These active and needed cells, in turn, dictate which key bits must be guessed during the key recovery step.

Entropy of the key bits This is arguably the most challenging aspect and remains an open problem. When the key schedule is a permutation of bits or can be described with simple rules, as in the case of `SKINNY`, the modeling is relatively straightforward. However, even for linear key schedules, it can become complex to enumerate all possible relations, including key-bridging relations, and no systematic approach currently exists.

To address this limitation, users have two main options. They can either assume no relations exist between the key bits involved in the attack or artificially reduce the entropy estimated by the model and validate this assumption a posteriori.

Computing the complexity Given an n-bit cipher with m-bit cells and a key size of κ bits, we propose the following constraints to describe the complexity of the attack. Let us assume the model already set the variable p as the ($-\log_2$ of the) probability for a pair to satisfy the input and output of the impossible differential, the variable k as the ($-\log_2$ of the) entropy of $k_{in} \cup k_{out}$ and both the variables D_{in} and D_{out} representing the dimension of the active cells in the plaintext and the ciphertext respectively.

The first step is to model the data complexity D as well as the number of pairs N. This can be achieved with the following constraints:

$$\begin{cases} e \text{ binary variable}, \quad D_{max} \text{ continuous variable in } [0, n] \\ g \text{ continuous variable in } [1, m], \quad D \text{ continuous variable in } [0, n] \\ D_{max} \leq D_{in} + n \times e \\ D_{max} \leq D_{out} + n \times (1 - e) \\ N = p - 0.53 + g \\ D \geq (N + n + 1 - D_{max})/2 \\ D \geq N + n + 1 - D_{in} - D_{out} \end{cases}$$

If one tries to minimize D, he will reach the formula of the data complexity given in Section 2.1. The variable g represents the $-\log_2$ of the factor by which the key bits involved in the attack will be reduced and $2^{-0.53} \approx \ln(2)$. Then the time complexity of the attack can be constrained with the following inequalities:

$$\begin{cases} T \text{ continuous variable in } [0,\kappa], f \text{ binary variable} \\ T \geq D \\ T \geq N + c'_E \\ T \geq k - p + N + c'_E \\ T \geq \kappa - g - n \times f \\ g \geq m \times (1 - f) \end{cases}$$

The last two constraints are a bit different than the complexity formula given by Boura *et al.* in [10]. Here we propose to limit the reduction factor g to m-bit, since it is often more efficient to perform a second attack requiring one more guess than brute-forcing the remaining keys. But obviously, the constraints can be modified, depending on the target.

3.3 Validator model

The validator model is a rather classical model searching for differential characteristics but for which both the input and output activity patterns are given. It only deals with the r_d rounds of the distinguishers. If the model is unsatisfiable, it implies that the activity patterns are incompatible, leading to the discovery of an impossible differential distinguisher. While, in theory, the validator model does not require the same granularity as the generator model, it would be unusual to adopt a different level of precision. In all applications described in Section 4, we used the same granularity for both models.

That said, the validator model differs from the generator model in its treatment of 1's. Although we decided to disregard them within the generator model, the validator model must account for them. Specifically, for each $*$ in Δ_X and Δ_Y we must decide whether it remains a $*$ or is set to 1 for the differential to be impossible. This consideration is particularly important in bit-oriented models, since the number of 1's directly affects the dimensions of Δ_X and Δ_Y which may, in turn, affect the time complexity of the associated attack.

A secondary goal of the validator model can therefore be to maximize $|\Delta_X| + |\Delta_Y|$. To achieve this, whenever the model finds a valid differential, we impose a new constraint that forces at least one of the $*$ bits previously set to 0 in either the input or output to be set to 1 and rerun the model. Asking the model to maximize the number of 1's will make the strategy highly effective. Finally, if the model eventually becomes unsatisfiable, we execute a simple model containing only the extra constraints previously added with the objective to minimize the number of 1's.

4 Applications

Our code was written using either the Python API or the C API for Gurobi. The computations run on a standard laptop, taking only seconds for the simplest cases and a few minutes for more complex ones.

We provide visualizations of some of the best impossible differential (ID) distinguishers across all applications, using a straightforward color-coding scheme. A red triangle in the upper-left corner of a cell (◸) indicates an unknown bit or cell in the forward trail, while a blue triangle in the lower-right corner (◢) represents an unknown bit or cell in the backward trail. Cells outlined in green correspond to the bits or cells used to deduce new zeros in the summary trail.

4.1 Application to AES

The AES (Advanced Encryption Standard) is a block cipher with a block size of 128 bits. It comes in three variants, differing in key sizes—128, 192, or 256 bits—and the corresponding number of encryption rounds. The block state is represented as a 4×4 array of bytes. Encryption begins with an initial round-key addition, followed by multiple iterations of the same round function, which is identical across all AES versions. The round function consists of four byte-oriented transformations:

– **SubBytes (SB):** Applies the same S-box substitution to each byte of the state.
– **ShiftRows (SR):** Rotates the second, third, and fourth rows of the state to the left by one, two, and three bytes, respectively.
– **MixColumns (MC):** Multiplies each column of the state by a Maximum Distance Separable (MDS) matrix.
– **AddRoundKey (ARK):** XORs the state with the current round key.

In the final round, the **MixColumns** operation is omitted.

Modeling the distinguisher We use the arbitrary S-box model, and the propagation through the linear layers is encoded using the simplified constraints for modeling MDS matrices, as described in Section 3. Let X_f, X_b, and X_s denote arrays of $r_{\mathrm{d}} \times 16$ binary variables representing the states immediately before the **SB** operation for the forward, backward, and summary trails, respectively. For all r and i, the following inequalities hold:

$$2 \times X_s[r][i] \le X_f[r][i] + X_b[r][i] \le X_s[r][i] + 1,$$

relating these three trails and ensuring that a state byte in the summary trail is zero if and only if it is zero in either the forward or backward trails.

The propagation in the forward trail between rounds r and $r+1$ is encoded with only a few simple constraints. First, after MixColumns, a column is either fully active or fully inactive which can be translated into the constraints:

$$X_f[r+1][4i] = X_f[r+1][4i+j] \quad \text{for all } 0 \le j < 4.$$

And then it is fully active if and only if at least one of the 4 cells involved in the MixColumns is active:

$$X_f[r+1][4i] \le \sum_{j=0}^{3} X_f[r][4(i+j \mod 4)+j] \le 4 \times X_f[r+1][4i].$$

Note that we combined here in one step the MixColumns and ShiftRows operations and that at the cell level, the SubBytes and AddRoundKey can be skipped as they do not alter the activity pattern.

Modeling the key-recovery Let X_{in} be an array of $r_{\text{in}} \times 16$ binary variables representing the states immediately before the **SB** operation in the extension of the distinguisher to the plaintext. The end of X_{in} must match the input of the distinguisher, resulting in the constraint $X_{\text{in}}[r_{\text{in}}] = X_f[0]$. The transition between $X_{in}[r_{\text{in}}]$ and $X_{in}[r_{\text{in}}-1]$ does not need to be deterministic since we can add any linear relation between the active bytes at the input of the distinguisher. Using the MDS property, the constraints between these two states are given as:

$$\begin{cases} e_{r_{\text{in}},i} \text{ is a binary variable,} \\ \sum_j X_{\text{in}}[r_{\text{in}}-1][4(i+j \mod 4)+j] + X_{\text{in}}[r_{\text{in}}][4i+j] \geq 5e_{r_{\text{in}},i}, \\ \sum_j X_{\text{in}}[r_{\text{in}}-1][4(i+j \mod 4)+j] + X_{\text{in}}[r_{\text{in}}][4i+j] \leq 8e_{r_{in},i}. \end{cases}$$

For the remaining rounds, propagation must occur with probability 1 up to the plaintext, so the constraints are identical to those for the backward trail. To compute the probability of a pair of plaintexts reaching the distinguisher's input, we define a variable $e_{r,i}$ for each column. The $-\log_2$ of the transition probability $p_{r,i}$ is then computed as:

$$p_{r,i} = 8(4e_{r,i} - \sum_j X_{\text{in}}[r][4i+j]), \quad \text{for } r \geq 1.$$

The dimension of the difference in the plaintext, D_{in}, is computed as $D_{\text{in}} = \sum_i X_{\text{in}}[0][i]$. Finally, the entropy k of the key material required for key recovery is obtained by counting the number of active variables in X_{in}, excluding $X_{\text{in}}[r_{\text{in}}]$. For each equation in the key schedule, if all variables in the equation are active, we subtract 1 from k. Note that our model includes only the original equations of the key schedule and does not account for advanced relations such as key bridging. A similar set of constraints can be derived for the extension of the distinguisher to the ciphertext.

Result Our model efficiently identified the best impossible attack on `AES-128`, producing results instantly. The output matched the attack described by Mala *et al.* [27], which corresponds to the basic version of the best-known impossible differential attack against 7-round `AES-128`, as later refined by Leurent and Pernot [24]. Specifically, Leurent and Pernot enhanced the attack by leveraging multiple distinguishers and replacing the exhaustive search step with a clever procedure relying on their innovative representation of the `AES` key schedule.

4.2 Application to `SKINNY`

`SKINNY` is a family of tweakable block ciphers introduced in [3]. It is based on a Substitution-Permutation Network (SPN) structure. Each instance is denoted as

SKINNY-n-zn, where n represents the block size (either 64 or 128 bits), and zn refers to the tweakey size, with $z \in \{1, 2, 3\}$. The state is composed of 16 cells, each of size $c \in \{4, 8\}$ bits. The round function consists of four main operations, similar to the ones of AES: the **SubCell (SC)** layer which applies the same S-box on each cell of the state, the addition of round constants and $8c$-bit round tweakeys, a **ShiftRows (SR)** operation which is actually the inverse of the AES one, and a **MixColumns (MC)** operation which involves a non-MDS binary matrix.

Impossible differential cryptanalysis was included in the security analysis conducted by the designers. Using the *miss-in-the-middle* technique, they discovered distinguishers spanning 11 rounds and developed a 16-round attack on SKINNY-n-n based on this distinguisher. Subsequent research proposed attacks on SKINNY-n-n, SKINNY-n-$2n$, and SKINNY-n-$3n$ in both single-tweakey and related-tweakey settings [17,18,25,39,41]. In this work, we focused on the single-tweakey setting and aimed to recover the best known attacks, as described in [18].

Modeling the Distinguisher SKINNY is cell-oriented so we employed a cell-based model combined with the arbitrary S-box model. The generator model is tasked with identifying new zero cells in the only remaining complex layer, the MixColumns operation:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1\,0\,1\,1 \\ 1\,0\,0\,0 \\ 0\,1\,1\,0 \\ 1\,0\,1\,0 \end{pmatrix}}_{M} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

We exhaustively enumerated all possible patterns to identify those that result in new inactive cells. For instance, if $x_2 = y_2 = 0$, then $0 = y_2 = x_1 + x_2$, which implies $x_1 = 0$. Thus, the pattern $(*, *, 0, *, *, *, 0, *)$ leads to a new inactive cell (x_1). Using the Espresso logic minimizer, we derived compact constraints, yielding 31 inequalities.

Modeling the Key-Recovery The key-recovery part of our model is similar to the one used for AES, but specific adaptations are required due to the particularities of SKINNY's tweakey schedule and the non-MDS matrix of the **MC** operation. Notably, inactive cells may occasionally need to be used to compute the value of an active cell, which is required to verify whether the partially encrypted pair of plaintexts belongs to ΔX. To account for this, we introduce a new binary variable, *needed*, for each cell. This variable encodes whether the value of the corresponding cell is necessary during the key-recovery step. The rules for identifying *needed* cells are as follows:

- **Active cells:** If a cell is *active*, it is also *needed*.
- **Propagation through MixColumns:** If any of the output cells y_j is *needed*, all the input cells x_i contributing to y_j are also *needed*. This propagation corresponds to the transpose of the MixColumns matrix M.

– **Round-tweakey cells:** If a round-tweakey cell is required to compute a *needed* state cell, the round-tweakey cell is included in k_{in}.

To identify k_{out}, we apply similar rules but swap the **AddTweakey** and **MixColumns** operations. The entropy of $k_{\text{in}} \cup k_{\text{out}}$ can then be accurately computed by, for instance, using the constraints provided by Qin *et al.* in [29]. More precisely, SKINNY-n-zn's tweakey schedule is merely a permutation, and guessing a cell in z round-tweakeys is sufficient to uniquely determine its value across all other round-tweakeys.

Results For all variants of SKINNY, our model identifies optimal attacks in under 10 seconds, processing up to only five candidates suggested by the generator model. Using this approach, we successfully recover the three attacks described in [18] in the single-key setting. For SKINNY-n-n, our model identified three distinguishers leading to attacks with the same complexity, which is dominated by the data complexity (see Figure 7). Similarly, for SKINNY-n-$3n$, two distinct attacks with optimal complexity were found. One attack involves 42 round-tweakey bits, while the other involves 43 round-tweakey bits.

4.3 Recovering the distinguishers of [33] on Midori-128

Midori [1] is a lightweight block cipher designed to minimize energy consumption during execution. It has two versions: Midori-64 (with a 64-bit block size) and Midori-128 (with a 128-bit block size). Since Midori-64 has been fully broken (see [6, 16, 38]), we focus on Midori-128.

The cipher uses four different 8-bit S-boxes, each defined as follows: a bit permutation p_i, followed by the parallel application of two 4-bit S-boxes, and finally the inverse bit permutation p_i^{-1}. A group of 8 bits entering an 8-bit S-box is referred to as a *cell*, while the cell's bits are divided into two *subcells*, processed by 4-bit S-boxes:

– The **upper subcell** consists of the 4 bits mapped by p_i to positions $\{0, \ldots, 3\}$.
– The **lower subcell** consists of the 4 bits mapped to positions $\{4, \ldots, 7\}$.

The first impossible differential attack on Midori was introduced in 2016 by Chen and Wang [43], using a 6-round distinguisher with two equal differences in the input pattern. Sasaki and Todo [33] later proposed 12 weight-1 distinguishers for 7 rounds of Midori-128. They also demonstrated that no weight-1 distinguishers exist for 8 rounds when the 4-bit S-boxes are treated as arbitrary.

Models Due to the structured design of the 8-bit S-boxes, we implemented the generator model at the bit level, utilizing an arbitrary S-box model for the underlying 4-bit S-boxes. New zeros were only searched around the **MixColumns** operation, as described in Section 3.

The validator model used the same granularity as the generator model. Interestingly, we observed that adding redundant constraints for the inverse **MC** operation (*i.e.* since the matrix is involutive, it means swapping the input and output variables in the inequalities) significantly accelerated the solver—reducing runtime by approximately tenfold.

Results We used Gurobi's `PoolSearchMode` option to exhaustively search for all ID distinguishers over 7 rounds of `Midori-128`. Without optimization, the model required 8 minutes to find all the 1,513 candidates, 36 of which were valid ID distinguishers. Recognizing that a complete 4-bit S-box must be inactive to produce an ID distinguisher, we added a constraint requiring at least four new zeros to be deduced from the summary trail. With this optimization, the same 36 distinguishers were identified in under 5 minutes, involving exactly 1,200 calls to the validator model.

The results are as follows: Among the 36 distinguishers, 12 correspond to the weight-1 distinguishers described in [33]. The remaining 24 are slight variations of these distinguishers.

Let us give more insight about the 12 original distinguishers. For any $i \in \{1, 4, 5, 8, 9, 12\}$, an input pattern where all cells are inactive except the **upper** (resp. **lower**) subcell of cell i is incompatible with an output pattern where all cells are inactive except the **lower** (resp. **upper**) subcell of cell i. Now the additional 24 distinguishers arise from the following observation: for each of the 12 distinguishers, replacing the active subcell in the input (resp. output) pattern with a fully active cell (*i.e.* both subcells active) still results in an impossible differential, provided the change is not applied to both input and output. In summary:

– At the cell level, there are 6 distinct ID patterns. Each pattern activates the same cell in both the input and output patterns.
– For each of these patterns, there are 6 valid subcell activation combinations, all leading to ID distinguishers.

An example of an ID distinguisher with both subcells active in an input cell is shown in Figure 2.

Finally, the solver required only 2 minutes to confirm the absence of ID distinguishers over 8 rounds. Specifically, it demonstrated that no activity pattern exists that could lead to the deduction of new inactive cells.

Other kinds of SPNs such that the bit-oriented `PRESENT` and the LS-design low-latency `ARADI` can also be modeled in our framework. We give more details about this modeling in the appendix A and B and will focus here on bit-oriented Feistel networks: AndRX designs such as `SIMON` and ARX designs such as `SPECK`.

4.4 Application to AndRX and ARX designs

`SIMON` The `SIMON` family of blockciphers is a NSA design introduced in [2]. It is a AndRX design based on Feistel structure. Let us call L^i_j (resp R^i_j) bit i of the left (resp. right) branch entering the round j. Let T^i be the result of the application of the non-linear function on L^i. All the registers have the same size n. Then, the round function works as follows

$$\begin{cases} R^{i+1}_j = L^i_j \\ T^i_j = L^i_{(j+8) \bmod n} L^i_{(j+1) \bmod n} \oplus L^i_{(j+2) \bmod n} \\ L^{i+1}_j = K^i_j \oplus R^i_j \oplus T^i_j. \end{cases}$$

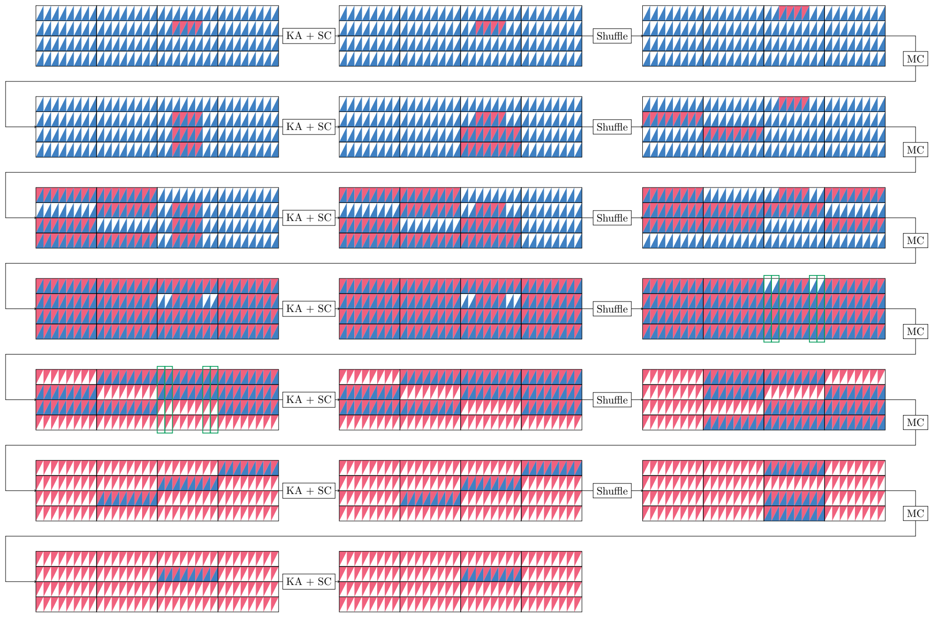


Fig. 2: Example of a new ID distinguisher on 7 rounds of MIDORI-128

SIMON has been the object of much scrutiny since its publication. Regarding impossible differential cryptanalysis, the first attacks were exposed by Boura *et al.* in [10]. Other distinguishers and attacks were manually obtained [12, 23, 36] and then automatically [14, 33]. Note that Sasaki and Todo also proved the absence of impossible differential distinguisher of weight 1 on 12 rounds [33] of SIMON-32. More recently, Zhang *et al.* proved that the length of the known impossible differential distinguishers are optimal, even when considering indirect contradictions [44], but in 2024, Chakraborty *et al.* used their CP tool to improve the previously known attacks by at least one round on all versions [11].

Modeling SIMON is bit-oriented cipher so is our model. New zeros can be obtained at two places through the round function. First if exactly two differences are null among the 3 bits L_j^{i+1}, R_j^i and T_j^i then the difference in the third one has to be null as well. The corresponding constraints are thus:

$$\begin{cases} L_j^{i+1} + R_j^i + T_j^i \geq z \\ L_j^{i+1} + R_j^i + T_j^i \leq 3 - 2z \end{cases}$$

Figure 3 zooms on the fourth round of a impossible differential distinguisher for SIMON-32 to give an example of the deduction of a new zero. The model detects a new zero around the XOR of bit number 8 of T^4 and R^4. Second, if the difference in $L^i_{(j+8) \bmod n}$ and $L^i_{(j+1) \bmod n}$ are null as well as the difference in either T^i_j or $L^i_{(j+2) \bmod n}$ then the difference in both T^i_j and $L^i_{(j+2) \bmod n}$ is

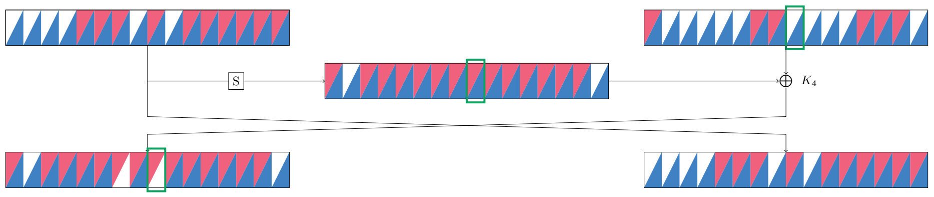


Fig. 3: New zeros in a Feistel Structure like SIMON.

null. This can be modeled by:

$$\begin{cases} L^i_{(j+8) \bmod n} + L^i_{(j+1) \bmod n} \leq 2(1-z) \\ T^i_j + L^i_{(j+2) \bmod n} \geq z \\ T^i_j + L^i_{(j+2) \bmod n} \leq 2 - z \end{cases}$$

Finally, we add a constraint to deal with the rotation invariance of the state by forcing the first bit of either the left or the right branch to be active: $L^0_0 + R^0_0 \geq 1$.

Results We recovered distinguishers of the same length as the previously known in less than 5 seconds for each version of `SIMON`. The number of candidates sent to the validator model until finding a valid ID was at most 20. We were also able to prove the absence of impossible differential for larger numbers of rounds in less than an hour on our laptop.

We can also include in our model the key recovery steps. Our modeling is looser than the one of [11] so the complexity bounds are less tight: contrary to [11], we cannot precisely define the filtering bits and therefore preferred to use the naive estimate of c_{in} as $D_{\text{in}} - D_X$ (respectively $c_{out} = D_{\text{out}} - D_Y$). Yet, we still obtain satisfactory results of attacks in a few minutes. For instance, we can reproduce the attack on 19 rounds of `SIMON-32-64` and also find one using fewer key bits (see Figure 6) for a total complexity of $2^{43.47}$. Similarly, we obtain attacks for all variants of `SIMON` in minutes but we needed to set up a time limit of 15 minutes in order to cut off long optimality proofs.

`Simeck` We defer the analysis of `Simeck`, really close to the one of `SIMON` to the appendix C.2.

`SPECK` The `SPECK` family of blockciphers has been introduced with SIMON in [2]. It is an ARX design based on Feistel structure. Let us call L^i (resp R^i) the left (resp. right) branch entering the round i. All the registers have the same size n. Then, the round function works as follows

$$\begin{cases} L^{i+1} = ((L^i \lll \alpha) \boxplus R^i) \oplus K^i \\ R^{i+1} = (R^i \ggg \beta) \oplus L^{i+1} \end{cases}$$

Modeling Obviously we used a bit-based modeling, mostly relying on the *full adder*, the function $f(x, y, c) = (x \oplus y \oplus c, xy \oplus cx \oplus cy)$, which can be used to iteratively compute the modular addition (see [11] for more details). We mainly considered it as an S-box and apply the same modeling techniques than for the other ciphers. The new zeros can be created around the XORs and the full adders. To simplify the problem, we removed the XOR of both branches in the last round since it is a linear operation.

Results We were able to prove that there are no ID distinguisher on 7 rounds instantaneously and for all versions of `SPECK`. It also took only few seconds to find ID distinguishers on 6 rounds for all versions as well.

5 Conclusion and future works

This work mainly brings simplification in the field of automatic search for impossible differential distinguishers and attacks. This simplification comes with two main side advantages. The first is that it is really simple to get familiar with our model and thus we hope that many designers and cryptanalysts will apply it to various cryptographic algorithms. The second improvement is the solving time: a generic solver finds a solution nearly instantaneously for most of the applications we demonstrated.

Throughout our work, we noticed several areas for improvement in our model. The first limit is that we only considered the deduction of new zeros in a fixed position in the generator model. However, other behaviours could indicate a probable impossible differential transition. For example, if the dimension of the set of solutions to a $0/*$ pattern is smaller than the dimension of the pattern, it indicates that some zeros will appear in any instantiation of the pattern. Yet, taking into account this wider class of deduction may lead to too many false candidates flooding the validator model. Moreover, we are also convinced that the key recovery steps of our generator model could be adapted to the probabilistic propagation framework [34]. On a wider perspective, most of the previous works proposed both a model for zero-correlation cryptanalysis and for impossible differential cryptanalysis because of their duality. We did not demonstrate the ability of our model to detect the former but at least for the distinguisher part, it should be a straightforward application using the dual structures of the ciphers.

References

1. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 411–436. Springer, Berlin, Heidelberg (Nov / Dec 2015). `https://doi.org/10.1007/978-3-662-48800-3_17`
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013), `https://eprint.iacr.org/2013/404`

3. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 123–153. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_5
4. Bellini, E., Formenti, M., Gérault, D., Grados, J., Hambitzer, A., Huang, Y.J., Huynh, P., Rachidi, M., Rohit, R., Tiwari, S.K.: Claasping ARADI: automated analysis of the ARADI block cipher. In: Mukhopadhyay, S., Stanica, P. (eds.) Progress in Cryptology - INDOCRYPT 2024 - 25th International Conference on Cryptology in India, Chennai, India, December 18-21, 2024, Proceedings, Part II. Lecture Notes in Computer Science, vol. 15496, pp. 90–113. Springer (2024), https://doi.org/10.1007/978-3-031-80311-6_5
5. Bellini, E., Huynh, P., Gerault, D., Visconti, A., Piccoli, A.D., Pelizzola, S.: Impossible differential automation: Model generation and new techniques. Cryptology ePrint Archive, Paper 2024/1998 (2024), https://eprint.iacr.org/2024/1998
6. Beyne, T.: Block cipher invariants as eigenvectors of correlation matrices. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11272, pp. 3–31. Springer (2018), https://doi.org/10.1007/978-3-030-03326-2_1
7. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 12–23. Springer, Berlin, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_2
8. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Berlin, Heidelberg (Sep 2007). https://doi.org/10.1007/978-3-540-74735-2_31
9. Boura, C., Lallemand, V., Naya-Plasencia, M., Suder, V.: Making the impossible possible. Journal of Cryptology **31**(1), 101–133 (Jan 2018). https://doi.org/10.1007/s00145-016-9251-7
10. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 179–199. Springer, Berlin, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45611-8_10
11. Chakraborty, D., Hadipour, H., Nguyen, P.H., Eichlseder, M.: Finding complete impossible differential attacks on andrx ciphers and efficient distinguishers for ARX designs. IACR Trans. Symmetric Cryptol. **2024**(3), 84–176 (2024), https://doi.org/10.46586/tosc.v2024.i3.84-176
12. Chen, Z., Wang, N., Wang, X.: Impossible differential cryptanalysis of reduced round SIMON. Cryptology ePrint Archive, Report 2015/286 (2015), https://eprint.iacr.org/2015/286
13. Derbez, P.: Note on impossible differential attacks. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 416–427. Springer, Berlin, Heidelberg (Mar 2016). https://doi.org/10.1007/978-3-662-52993-5_21
14. Derbez, P., Fouque, P.A.: Automatic search of meet-in-the-middle and impossible differential attacks. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II.

LNCS, vol. 9815, pp. 157–184. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_6
15. Greene, P., Motley, M., Weeks, B.: ARADI and LLAMA: low-latency cryptography for memory encryption. IACR Cryptol. ePrint Arch. p. 1240 (2024), https://eprint.iacr.org/2024/1240
16. Guo, J., Jean, J., Nikolic, I., Qiao, K., Sasaki, Y., Sim, S.M.: Invariant subspace attack against midori64 and the resistance criteria for s-box designs. IACR Trans. Symmetric Cryptol. **2016**(1), 33–56 (2016), https://doi.org/10.13154/tosc.v2016.i1.33-56
17. Hadipour, H., Gerhalter, S., Sadeghi, S., Eichlseder, M.: Improved search for integral, impossible differential and zero-correlation attacks application to Ascon, Fork-SKINNY, SKINNY, MANTIS, PRESENT and QARMAv2. IACR Trans. Symm. Cryptol. **2024**(1), 234–325 (2024). https://doi.org/10.46586/tosc.v2024.i1.234-325
18. Hadipour, H., Sadeghi, S., Eichlseder, M.: Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part IV. LNCS, vol. 14007, pp. 128–157. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30634-1_5
19. Hu, K., Peyrin, T., Wang, M.: Finding all impossible differentials when considering the DDT. In: Smith, B., Wu, H. (eds.) SAC 2022. LNCS, vol. 13742, pp. 285–305. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-58411-4_13
20. Hu, X., Jiao, L.: Pre-sieve, partial-guess, and accurate estimation: Full-round related-key impossible boomerang attack on ARADI. Cryptology ePrint Archive, Paper 2025/056 (2025), https://eprint.iacr.org/2025/056
21. Kim, J., Hong, S., Sung, J., Lee, C., Lee, S.: Impossible differential cryptanalysis for block cipher structures. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 82–96. Springer, Berlin, Heidelberg (Dec 2003). https://doi.org/10.1007/978-3-540-24582-7_6
22. Knudsen, L.: Deal-a 128-bit block cipher. complexity **258**(2), 216 (1998)
23. Kondo, K., Sasaki, Y., Iwata, T.: On the design rationale of simon block cipher: Integral attacks and impossible differential attacks against simon variants. In: Manulis, M., Sadeghi, A.R., Schneider, S. (eds.) ACNS 16International Conference on Applied Cryptography and Network Security. LNCS, vol. 9696, pp. 518–536. Springer, Cham (Jun 2016). https://doi.org/10.1007/978-3-319-39555-5_28
24. Leurent, G., Pernot, C.: New representations of the AES key schedule. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 54–84. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_3
25. Liu, G., Ghosh, M., Song, L.: Security analysis of SKINNY under related-tweakey settings (long paper). IACR Trans. Symm. Cryptol. **2017**(3), 37–72 (2017). https://doi.org/10.13154/tosc.v2017.i3.37-72
26. Luo, Y., Lai, X., Wu, Z., Gong, G.: A unified method for finding impossible differentials of block cipher structures. Inf. Sci. **263**, 211–220 (2014), https://doi.org/10.1016/j.ins.2013.08.051
27. Mala, H., Dakhilalian, M., Rijmen, V., Modarres-Hashemi, M.: Improved impossible differential cryptanalysis of 7-round AES-128. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 282–291. Springer, Berlin, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17401-8_20
28. McCluskey, E.J.: Minimization of boolean functions. The Bell System Technical Journal **35**(6), 1417–1444 (1956)

29. Qin, L., Dong, X., Wang, X., Jia, K., Liu, Y.: Automated search oriented to key recovery on ciphers with linear key schedule. IACR Trans. Symm. Cryptol. **2021**(2), 249–291 (2021). https://doi.org/10.46586/tosc.v2021.i2.249-291
30. Sadeghi, S., Bagheri, N.: Improved zero-correlation and impossible differential cryptanalysis of reduced-round simeck block cipher. IET Information Security **12**(4), 314–325 (2018)
31. Sadeghi, S., Bagheri, N.: Improved zero-correlation and impossible differential cryptanalysis of reduced-round SIMECK block cipher. IET Inf. Secur. **12**(4), 314–325 (2018), https://doi.org/10.1049/iet-ifs.2016.0590
32. Sadeghi, S., Bagheri, N.: Security analysis of simeck block cipher against related-key impossible differential. Information Processing Letters **147**, 14–21 (2019)
33. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 185–215. Springer, Cham (Apr / May 2017). https://doi.org/10.1007/978-3-319-56617-7_7
34. Song, L., Yang, Q., Chen, Y., Hu, L., Weng, J.: Probabilistic extensions: A one-step framework for finding rectangle attacks and beyond. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 339–367. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58716-0_12
35. Sun, B., Liu, M., Guo, J., Rijmen, V., Li, R.: Provable security evaluation of structures against impossible differential and zero correlation linear cryptanalysis. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 196–213. Springer, Berlin, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49890-3_8
36. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Constructing mixed-integer programming models whose feasible region is exactly the set of all valid differential characteristics of SIMON. Cryptology ePrint Archive, Report 2015/122 (2015), https://eprint.iacr.org/2015/122
37. Tezcan, C.: Improbable differential attacks on present using undisturbed bits. J. Comput. Appl. Math. **259**, 503–511 (2014), https://doi.org/10.1016/j.cam.2013.06.023
38. Todo, Y., Leander, G., Sasaki, Y.: Nonlinear invariant attack - practical attack on full SCREAM, iSCREAM, and Midori64. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 3–33. Springer, Berlin, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_1
39. Tolba, M., Abdelkhalek, A., Youssef, A.M.: Impossible differential cryptanalysis of reduced-round SKINNY. In: Joye, M., Nitaj, A. (eds.) AFRICACRYPT 17. LNCS, vol. 10239, pp. 117–134. Springer, Cham (May 2017). https://doi.org/10.1007/978-3-319-57339-7_7
40. Wu, S., Wang, M.: Automatic search of truncated impossible differentials for word-oriented block ciphers. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 283–302. Springer, Berlin, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34931-7_17
41. Yang, D., Qi, W., Chen, H.: Impossible differential attacks on the SKINNY family of block ciphers. IET Inf. Secur. **11**(6), 377–385 (2017), https://doi.org/10.1049/iet-ifs.2016.0488
42. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The simeck family of lightweight block ciphers. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 307–329. Springer, Berlin, Heidelberg (Sep 2015). https://doi.org/10.1007/978-3-662-48324-4_16

43. Zhan, C., Xiaoyun, W.: Impossible differential cryptanalysis of midori. Cryptology ePrint Archive, Report 2016/535 (2016), `https://eprint.iacr.org/2016/535`
44. Zhang, K., Wang, S., Lai, X., Wang, L., Guan, J., Hu, B., Shi, T.: Impossible differential cryptanalysis and a security evaluation framework for AND-RX ciphers. IEEE Trans. Inf. Theory **70**(8), 6025–6040 (2024), `https://doi.org/10.1109/TIT.2023.3292241`

A Application to the bit-oriented cipher `PRESENT`

`PRESENT` is a 64-bit lightweight blockcipher introduced by Bogdanov *et al.* in [8]. Its round function is composed of a S-box layer followed by a permutation of the state bits and a key addition.

Modeling the Distinguisher Tezcan [37] provided ID distinguishers on 5 rounds in an arbitrary S-box model and on 6 rounds in the specific S-box model. The linear layer being a permutation, new zeros can only be obtained around an S-box. We applied the algorithm presented in Section 3 and found 42 possible patterns leading to new zeros. There is only one case where the deduction of a new zero occurs without completely inactivating the S-box: if we have the pattern $(*00*, *0*0)$ then we can deduce that the first bit of the output difference is actually zero. Regarding the validator model, we decided to do the more complete model possible by encoding the exact DDT in it.

We faced our first challenge in reproducing these results without considering the propagation of 1's in the generator model. The main problem is that the 6-round distinguishers requires the transition $(1001) \rightarrow (***0)$ through the S-box to be valid within the model. Unfortunately, the transition $(*00*) \rightarrow (***0)$ cannot be added to the model since it does not happen with probability one. Hence, there are two choices. The first one is to consider 1's as well is the generator model, making it more complicated. The second option is to artificially allow the transition even though it is wrong and to let the validator model handle it. This is the option we selected. However, we only allowed it in the first S-box layer (note that a similar transition is also valid for S^{-1} and we allowed it in the last S-box layer) to avoid too many false positive for the generator model.

Results Finding ID distinguishers on 6 rounds is instantaneous. The model decides also in less than one second that there is no ID in 7 rounds. Indeed, it does not find any candidate able to generate a new zero.

Interestingly, we were able to find a 6-round ID distinguisher such that all the S-boxes at the output are active. This heavy ID distinguisher is depicted in Figure 4. It contradicts one of the heuristic commonly used in the search of Impossible Differential distinguisher that the longest IDs are caused by pairs with very few active bits because otherwise inactive values would not be able to propagate with probability one to the central rounds of the distinguisher. In particular, we emphasize that the result of Sun *et al.* [35] about the reduction of any ID distinguisher to a low weight distinguisher only applies to arbitrary S-box models.

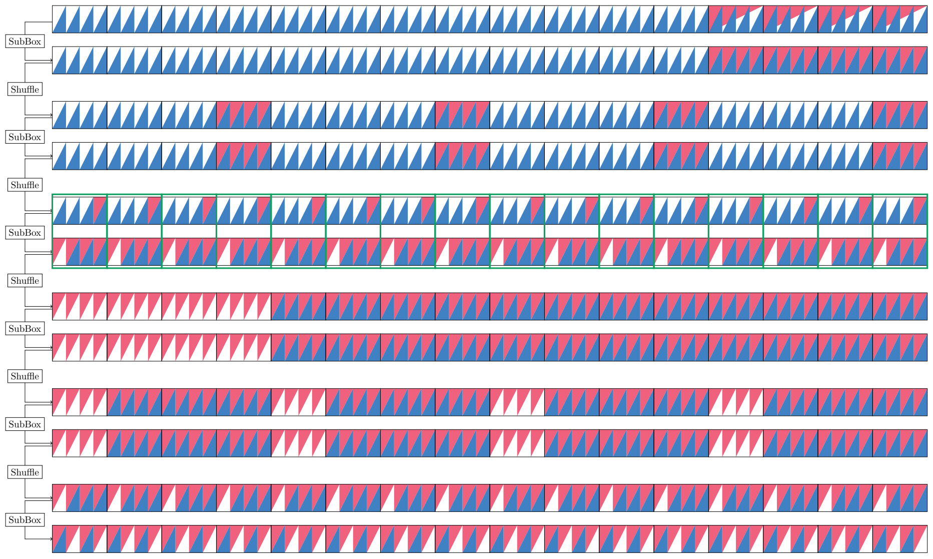


Fig. 4: A heavy ID distinguisher on 6 rounds of PRESENT (least significant bit on the left)

B Application to a new design ARADI

ARADI [15] is a low-latency blockcipher published by the NSA in 2024. It processes 128-bit blocks using 256-bit keys. An optimal 4-bit S-Box is used in combination with four different involutive linear layers Λ_i alternating every 4 rounds for a total of 16 rounds. In particular this means that partial-round security properties have to be evaluated regarding the number of rounds of this toy cipher and the index of the first linear layer used.

The first cryptanalysis of of ARADI, remarkably published only a few weeks after the design, has been provided by the automatic tool CLAASP [4] . It gives first results against various types of attacks. Regarding impossible differentials, it uses a validation approach (as in [33]) to demonstrate that any 7-round pattern leading from 1 active nibble to 1 active nibble is an impossible differential. It seems that it only considers partial-round security beginning at round 0. A recent preprint presents a full round related-key Impossible Boomerang attack [20]. However it seems unverified as it needs 2^{130} data.

We applied our framework at the nibble based on of ARADI. We were able to obtain the first impossible differential attack against 13 rounds represented in Figure 5. New zeros are obtained on the equations of the linear layer: sums of 4 nibbles having to be zeroed. Equations leading to new zeros are represented by nibbles being framed by the same color. We obtain $D_{\text{in}} = 4 \times 9 = 36$, $D_{\text{out}} = 4 \times 21 = 84$, $D_{\text{X}} = D_{\text{Y}} = 4$ and $|k_{\text{in}} \cup k_{\text{out}}| = 4 \times (16 + 39) = 220$. Finally, we get $c_{\text{in}} + c_{\text{out}} = 112$ and thus a data complexity of $c_{\text{in}} + c_{\text{out}} - 0.53 + n + 1 - |D_{\text{in}}| - |D_{\text{out}}| + g = 121.47$ (in log2) and a time complexity of $2^{220,47}$. Note that the last round does not completely appear in the figure as the missing step (S and Λ) will not change the dimension of the outer space.

The time complexity of the best 13-round attacks we obtain for other offsets are $2^{224.47}$ (for offset 1), $2^{252.47}$ (for offset 2) and $2^{240.47}$ (for offset 3). We did not found any attack on 14 rounds.

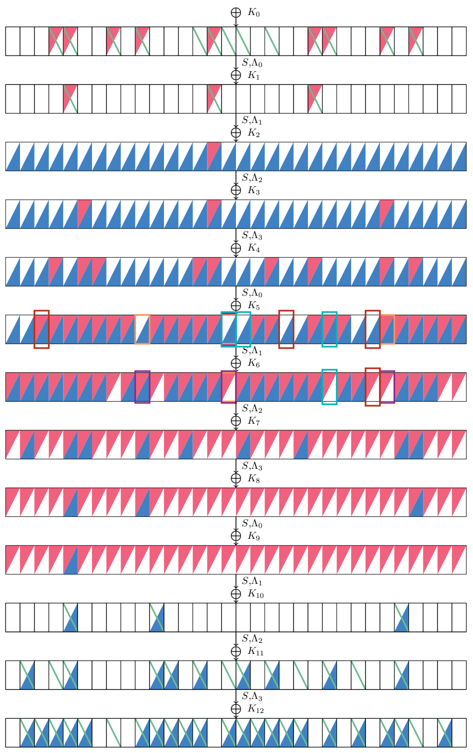


Fig. 5: The first ID attack on 13 rounds of `ARADI` beginning at round 0. Cells framed in the same colours are involved in the same deduction. The striped cells are the *needed* cells of the key recovery steps.

C More results on the AndRX SIMON and Simeck

C.1 SIMON

We also ran the solver with the objective to find impossible differential of heavy weights. We used the technique from Section 3.3 to deduce clusters from these distinguishers. For exemple, the distinguisher on 11 rounds of `SIMON-32`

$$0000000000000000\ 1000000100000010 \nrightarrow 1000000000000010\ 0000000000000000$$

is in fact the convex hull of the following impossible differential distinguisher cluster containing 16 characteristics

$$0000000000000000{*}0000001000000{*}0 \nrightarrow {*}0000000000000{*}00000000000000000.$$

Similarly, we obtained high dimensional clusters for all versions of `SIMON` in few seconds. The bigger one deals with 19 rounds of `SIMON-128` and has dimension 48:

$$\begin{array}{c}
0{*}0{*}000{*}{*}{*}{*}00{*}{*}{*}{*}{*}0{*}{*}{*}{*}{*}{*}{*}{*}0000000000000000000000000000000000000\\
{*}{*}{*}00{*}{*}{*}{*}{*}0{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}1{*}00000000000000000000000000000{*}0{*}000{*}\\
\not\downarrow\\
0000000000000000000000000{*}0{*}000000000000000000000000000000000000\\
0000000000000000000000000000000000000000000000000000000000000000
\end{array}$$

C.2 SIMECK

The `Simeck` family of blockciphers is an optimized version of `SIMON` introduced in [42]. It is structurally similar to `SIMON` except it uses a different non linear function :

$$T_j^i = L_j^i L_{(j+5) \bmod n}^i \oplus L_{(j+1) \bmod n}^i.$$

As for `SIMON`, we focused on the search of new zeros around the XOR of the Feistel structure and reduced the search space by forcing the first bit of either the left or the right branch to be active in the input pattern.

The designers of `Simeck` provided in their security analysis first estimations on the number of rounds that could be attacked by impossible differential distinguishers. However, it was only focused on direct contradiction between the forward and the backward trail. These attacks were later improved using indirect contradictions found manually ([31]) or automatically ([11, 44]).

We did the same modeling as for `SIMON`, with only the slight modification of the rotation coefficients. We tried two approaches to generate distinguishers. The first one consists in searching for candidates with a maximum number of newly deduced inactived cells in the generator model. Indeed, intuitively, these candidates will lead to impossible transitions with high probability and thus the validator model will have to verify less candidates. Remarkably, we recovered distinguishers of the same length than the previously known in seconds without the burden of the modelisation of the propagation of deductions usually needed to find indirect contradiction. We are also able to prove the absence of impossible differential for larger number of rounds in less than a minute. The other approach is to consider heavy weight candidates. Indeed, as seen in the section about `SIMON`, they are associated to big clusters of candidates. This approach takes longer but most of the solving time is used to prove the optimality of the solutions. The best solution is indeed found in a few seconds so the search could have been stopped before the end of the process. The performances of the search are given in Table 2.

Version	r_D	Generated candidates (Valid ones)	Time	Deduced inactive cells	Active bits
`Simeck`-32	11	7 (2)	0.8s	6	2
	12	805 (0)	37s	-	-
`Simeck`-48	15	8 (1)	1s	4	2
	16	22 (0)	2.7s	-	-
`Simeck`-64	17	25 (1)	5s	6	2
	18	184 (0)	30s	-	-

(a) Objective function : maximize the number of deduced inactive cells

Version	r_D	Generated candidates (Valid ones)	Time (Best solution)	Deduced inactive cells	Active bits
`Simeck`-32	11	39 (2)	2.6 s (2s)	3	11
`Simeck`-48	15	23 (2)	2.6 s (0s)	4	3
`Simeck`-64	17	5292 (1)	630s (0s)	4	6

(b) Objective function : Maximize the number of active bits

Table 2: Performances of the search for distinguishers on `Simeck`

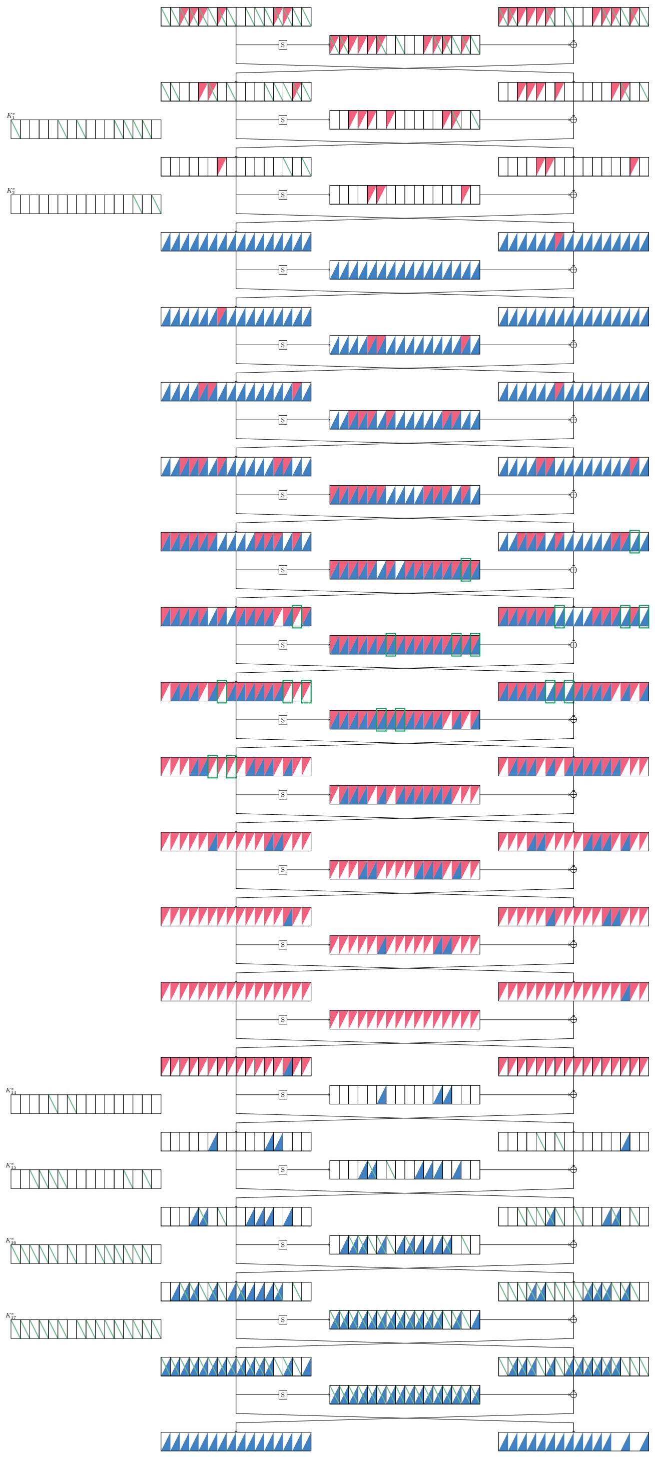
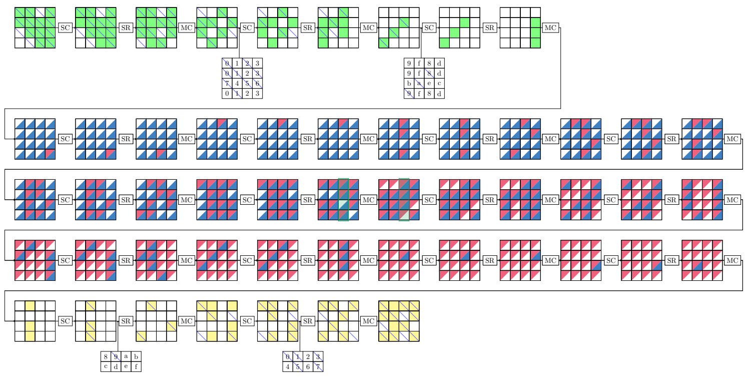
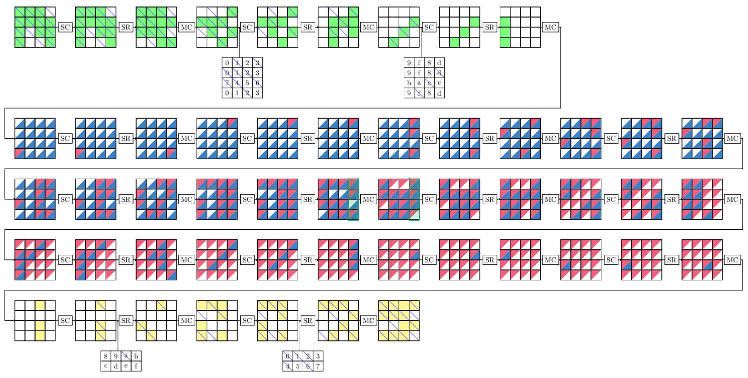


Fig. 6: A more efficient attack on 19 rounds of SIMON-32-64 than [11]. The striped cells are the *needed* cells of the key recovery steps.

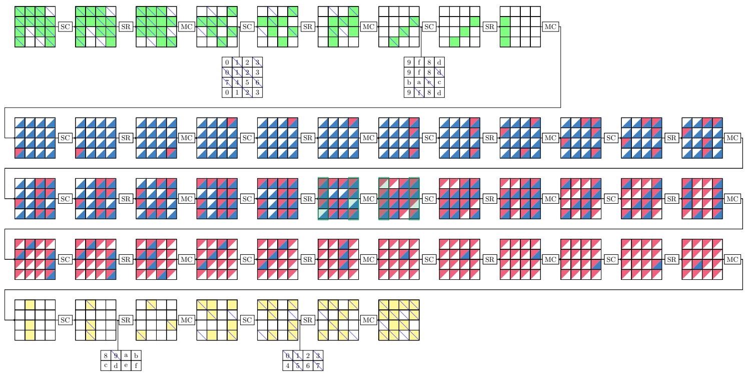
D Attacks on 17 rounds SKINNY-n-n



(a) $|k_{in} \bigcup k_{out}| = 10c$



(b) $|k_{in} \bigcup k_{out}| = 11c$



(c) $|k_{in} \bigcup k_{out}| = 12c$

Fig. 7: Three equivalent attacks on 17-round SKINNY-n-n. Green and yellow cells are the cells involved in the key recovery steps. In the three cases, they determine $c_{in} = c_{out} = 7c - c = 6c$.