New Models for the Cryptanalysis of ASCON

Mathieu Degré[1], Patrick Derbez[1], Lucie Lahaye[2], and André Schrottenloher[1]

[1] Univ Rennes, Inria, CNRS, IRISA
firstname.lastname@inria.fr
[2] ENS de Lyon
firstname.lastname@ens-lyon.fr

Abstract. This paper focuses on the cryptanalysis of the ASCON family using automatic tools. We analyze two different problems with the goal to obtain new modelings, both simpler and less computationally heavy than previous works (all our models require only a small amount of code and run on regular desktop computers).
The first problem is the search for Meet-in-the-middle attacks on reduced-round ASCON-Hash. Starting from the MILP modeling of Qin et al. (EUROCRYPT 2023 & ePrint 2023), we rephrase the problem in SAT, which accelerates significantly the solving time and removes the need for the "weak diffusion structure" heuristic. This allows us to reduce the memory complexity of Qin et al.'s attacks and to prove some optimality results.
The second problem is the search for lower bounds on the probability of differential characteristics for the ASCON permutation. We introduce a lossy MILP encoding of the propagation rules based on the Hamming weight, in order to find quickly lower bounds which are comparable to the state of the art. We find a small improvement over the existing bound on 7 rounds.

Keywords: ASCON, Symmetric Cryptanalysis, Meet-in-the-middle Attacks, Differential Cryptanalysis, Mixed Integer Linear Programming, SAT

1 Introduction

Ascon [8] is a family of permutation-based authenticated encryption and hash functions, which was selected in 2023 as the winner of the NIST Lightweight Encryption standardization process [22]. All functions in the Ascon family are based on the permutation of the same name, and use variants of Duplex [3] and Sponge [2] modes.

Being a high-profile target, Ascon has been the subject of more than 25 independent third-party cryptanalysis works to date, focusing on different aspects of the permutation and its modes, such as distinguishers, key-recovery, collision and preimage attacks on weakened variants. Our work targets both the permutation and the Ascon-Hash function.

Automatic Tools for Cryptanalysis. Automatic tools have been widely used in order to find and optimize attacks on the Ascon family, either reducing the search of an attack to a SAT, SMT or MILP problem [20] or an ad hoc problem which is solved automatically [15]. The main issue with Ascon is its large state size (320 bits) and weakly aligned structure, which essentially requires a model to define at least 320 variables for each round. The resulting SAT or MILP problems can be complex, computationally heavy to solve, and often do not terminate.

Contributions and Organization. In this work, we target two cryptanalytic problems with the aim of simplifying the models and reducing their runtime. The first problem is the optimization of Meet-in-the-middle preimage attacks on Ascon-Hash, which currently allow to attack the 3- or 4-round versions. Following a framework of Qin et al. [19,20], we design a simple SAT modeling (whereas they used MILP). The results with this new modeling are twofold: first, we are able to reduce significantly the memory footprint of the attacks (from 2^{54} to 2^{34} for the 4-round attack). Next, the SAT model allows us to prove that, under some hypotheses on the shape of the MITM attack path, this technique cannot reach 5 rounds, while the MILP modeling left this question opened.

The second problem is the search for differential characteristics of the permutation. We observe that a simple MILP model, with a lossy approximation of the differential transition table of the S-Box, allows to recover quite good lower bounds on the probabilities. This allows us to improve the current best lower bound on 7 rounds of Ascon.

The code of our models is available at:

https://gitlab.inria.fr/capsule/ascon-new-models

The paper is organized as follows. In Section 2, we recall the specification of the Ascon permutation and Ascon-Hash. In Section 3 we explain the MITM preimage attack framework of [19,20]. In Section 4 we explain our SAT modeling and its results. Finally in Section 5 we give our new model for differential bounds and results.

Related Work. The use of SAT in automated tools for Ascon is not new. In [10] a SAT modeling of propagations through the S-Box and linear layers, combined with cardinality constraints (which reuses ideas from [21]) was used to find bounds for differential and linear cryptanalysis. Our model is different since we target MITM attacks, but has some similarities.

In an independent line of work, Li et al. [17] optimized a different algebraic preimage attack using a SAT modeling as well. This led to small improvements in time complexity with respect to [20] and reduction of the memory to negligible. Therefore, the attacks that we present here are not strictly the best preimage attacks on Ascon-Hash, but the best MITM attack paths. The optimality results are also specific to the MITM setting.

2 Preliminaries

The primitives of the Ascon family are based on the Ascon permutation, which operates on a 320-bit state represented as an array of bits with 64 columns and 5 lines. Variants of the permutation are obtained by iterating the round function:

$$p = p_L \circ p_S \circ p_C$$

where p_L is the linear layer, p_S the S-Box layer, and p_C the addition of a constant. The n-round Ascon permutation is then simply written as p^n. These operations are defined as follows:

Constant addition – p_C XORs x_2 with a constant that depends on the round. This operation is insignificant for the analyses of this paper.

Substitution – p_S applies the 5-bit Ascon S-Box column-wise, as

$$S(y_0\ y_1\ y_2\ y_3\ y_4) = y'_0\ y'_1\ y'_2\ y'_3\ y'_4, \text{ where:}$$

$$\begin{cases} y'_0 = y_4y_1 \oplus y_3 \oplus y_2y_1 \oplus y_2 \oplus y_1y_0 \oplus y_1 \oplus y_0 \\ y'_1 = y_4 \oplus y_3y_2 \oplus y_3y_1 \oplus y_3 \oplus y_2y_1 \oplus y_2 \oplus y_1 \oplus y_0 \\ y'_2 = y_4y_3 \oplus y_4 \oplus y_2 \oplus y_1 \oplus 1 \\ y'_3 = y_4y_0 \oplus y_4 \oplus y_3y_0 \oplus y_3 \oplus y_2 \oplus y_1 \oplus y_0 \\ y'_4 = y_4y_1 \oplus y_4 \oplus y_3 \oplus y_1y_0 \oplus y_1 \end{cases} \quad (1)$$

Linear Layer – p_L transforms the rows $x_0, \dots, x_4$ into $x'_0, \dots, x'_4$, defined as follows, where $\ggg$ denotes a circular shift:

$$\begin{cases} x'_0 = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28) \\ x'_1 = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39) \\ x'_2 = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6) \\ x'_3 = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17) \\ x'_4 = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41) \end{cases} \quad (2)$$

2.1 The ASCON-Hash Function

The Ascon family defines several hash and extendable output functions (Ascon-Hash, Ascon-Hasha, Ascon-Xof, Ascon-Xofa). All use the same sponge construction with different parameters. The 320-bit state is divided into an *inner part*, of size c (the *capacity*), and an *outer part*, of size r (the *rate*). The input to the function is first padded and divided into blocks of size r. In the first phase (absorption) these blocks are XORed into the rate between two applications of the permutation. In the second phase (extraction) the current value of the rate is output between two applications of the permutation, until the wanted length is obtained. This process is illustrated in Figure 1[3]. The variant studied in this paper is Ascon-Hash, which uses $a = b = 12$, $r = 64$, and a hash size of $h = 128$ bits.

[3] This figure, like several in this paper, uses the "TikZ library for crypto" of [9].

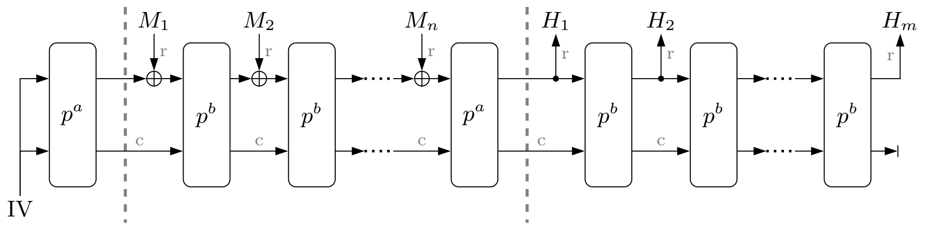


Fig. 1: Structure of Ascon hashing modes.

The padding scheme always consists in appending a “1” followed by as many zeroes as required to make the plaintext length a multiple of the rate. For the attacks on Ascon-Hash, we simply note that the last message block will contain at least 1 uncontrolled bit “1”.

3 Preliminaries on MITM Attacks

Meet-in-the-middle (MITM) attacks were introduced in the cryptanalysis of block ciphers by Diffie and Hellman [6], and were later extended to hash functions. Actually, most of the previous cryptanalysis results consider hash functions based on the Merkle-Damgård domain extension technique, using a compression function based on a block cipher. The techniques are therefore similar to the cryptanalysis of block ciphers [1,13].

In both cases, the idea of the MITM is to consider two subsets of internal states of the cipher (resp. compression function) which can be computed independently. The possibilities for these states are exhausted, and matching pairs of values are found, according to the points in which the two sub-paths meet.

The typical way is to split the computation in a *backward chunk* and a *forward chunk*. This works best when the full state can be known at the beginning of both chunks, for example when we start forwards from a plaintext and backwards from a ciphertext. However, this is not the case in a Sponge-based mode.

3.1 Meet-not-in-the-middle

At EUROCRYPT 2023, Qin et al. [19] introduced MITM-based preimage attacks on Ascon-Hash and Keccak, which were later improved in [20]. We will focus here on Ascon-Hash. The attack will focus on the second-to-last permutation call, which is reduced to R rounds instead of 12. The last message block M_3 is absorbed and the first hash block H_1 is returned. The goal is to find a block M_3 such that H_1 matches the target image. Afterwards, the attack is repeated until H_2 also matches. This situation is represented in Figure 2.

The idea of [19] is to separate the inner part at M_3 into three sets of bits: *red* bits 🟥, *blue* bits 🟦 and *gray* bits ⬛ (the remaining ones). Red and blue bits are the varying bits of what would have been previously the forward and backward

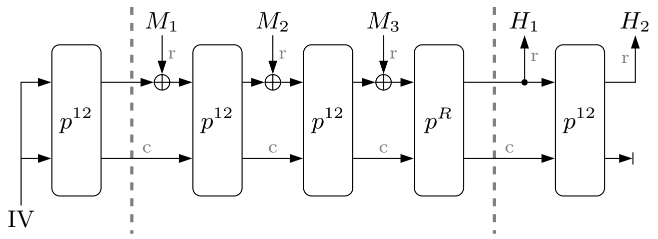


Fig. 2: Representation of the ASCON structure in the context of this attack

paths. Gray bits are fixed constants, including the padding bits which are always gray.

After fixing an initial red/blue/gray configuration, one analyzes the propagation of red and blue bits. The goal is that several bits of the inner part after p^R (*matching bits*) can still be expressed as a linear combination of: • a function of the red and gray bits; • a function of the blue and gray bits:

$$h = f(x_R, x_G) \oplus g(x_B, x_G) \oplus c \tag{3}$$

where c is some constant and x_R, x_B, x_G are the respective values of red, blue and gray bits.

Let h_t be the target value on the matching bits. Once such a configuration is found, we can compute a *MITM episode*. We take arbitrary messages (M_1, M_2) which will fix the internal state before p^R. For any value x_G:

1. Create a hash table U and for all x_R, put x_R in the table at index $f(x_R, x_G)$
2. For all x_B, compute $g(x_B) \oplus c \oplus h_t$ and search for matches in the table U. Any match gives a message block $M_3 = (x_B, x_R, x_G)$ such that H_1 has the value h_t at the matching bits.

For each such candidate, we need to recompute the entire hash value and check if it matches the desired image. The time (T) and memory (M) complexities of this episode can be expressed depending on the number of red bits $\mathcal{D}_r$, of blue bits $\mathcal{D}_b$ and matching bits $\mathcal{D}_m$. Counting in ASCON-Hash calls and considering only the dominating term in the complexity, we obtain a simple formula:

$$\begin{cases} T = 2^{63-\mathcal{D}_r-\mathcal{D}_b} \max\left(2^{\mathcal{D}_r}, 2^{\mathcal{D}_b}, 2^{\mathcal{D}_r+\mathcal{D}_b-\mathcal{D}_m}\right) \\ M = \min\left(2^{\mathcal{D}_r}, 2^{\mathcal{D}_b}\right) \end{cases} . \tag{4}$$

Here, $(63 - \mathcal{D}_r - \mathcal{D}_b)$ is the number of controllable gray bits, since we cannot control the last padding bit. In this complexity, $2^{\mathcal{D}_r}$ is the size of the *red list*, i.e., all entries $(x_R, f(x_R, x_G))$ of the table U; $2^{\mathcal{D}_b}$ is the size of the *blue list*, i.e., all entries $(x_B, g(x_B, x_G))$. Finally $2^{\mathcal{D}_r+\mathcal{D}_b-\mathcal{D}_m}$ is the size of the *merged list*, i.e., all matching pairs which require a recomputation of the hash. Notice that the blue and merged list are never actually stored in memory; furthermore, the roles of the blue and red list can be swapped.

However, this is not the entire attack yet, for two reasons: first, the message space of the block M_3 is of size 2^{63}, meaning that we can only match on average 63 bits of the target image, and we need to repeat this process 2^{65} times. Second, in order to facilitate the propagation of blue and red bits, it is better to enforce some constraints on the outer part before p^R. If there are N bits of constraints on the outer part, we need to compute it for 2^N random choices of M_1, M_2 before running the next MITM episode.

Therefore, the time and memory complexities of the full attack are computed as:

$$\begin{cases} T = \max\left(2^{65+N}, 2^{65} \times 2^{63-\mathcal{D}_r-\mathcal{D}_b} \max\left(2^{\mathcal{D}_r}, 2^{\mathcal{D}_b}, 2^{\mathcal{D}_r+\mathcal{D}_b-\mathcal{D}_m}\right)\right) \\ \quad\;\; = \max\left(2^{65+N}, 2^{128-\mathcal{D}_r}, 2^{128-\mathcal{D}_b}, 2^{128-\mathcal{D}_m}\right) \\ M = \min\left(2^{\mathcal{D}_r}, 2^{\mathcal{D}_b}\right) \quad . \end{cases} \tag{5}$$

3.2 Improvement with Cancellations

This first attack framework was improved by Qin et al. [19] using *bit cancellations*.

The reason a bit in the message block cannot be expressed as a linear combination of red and blue bits is the non-linearity introduced by the S-Box. If a red bit is multiplied by a blue one (in terms of logical AND) the result cannot be exploited anymore for the MITM.

To facilitate the propagation, it would be beneficial to have fixed the value of this blue bit (or the red one). This is the idea of *cancellations*. First, we select $\mathcal{A}_r$ red bits and $\mathcal{A}_b$ blue bits in the path, which we transform into gray bits. Then, we rewrite the functions f and g in terms of the values y_R and y_B taken at these bits:

$$f(x_R, x_G, y_B) \text{ and } g(x_B, x_G, y_R) \quad .$$

Notice that, by assumption, y_B is a function of the blue bits: $y_B = f'(x_B, x_G)$, and y_R a function of the red bits: $y_R = g'(x_R, x_G)$. Therefore, the original system of equations $h_t = f(x_R, x_G) \oplus g(x_B, x_G) \oplus c$ is transformed into a more complex system over the variables x_R, x_B, x_G, y_R, y_B:

$$\begin{cases} h_t = f(x_R, x_G, y_B) \oplus g(x_B, x_G, y_R) \oplus c \\ y_B = f'(x_B, x_G) \\ y_R = g'(x_R, x_G) \quad . \end{cases} \tag{6}$$

This amounts to introducing additional variables (y_R and y_B) to put this system of algebraic equations into MITM form. This technique can be compared to the linearization technique of [17], which would rather introduce new variables to make the system linear. More generally, cancellations can happen on any bit which is a linear expression of blue and red. We will say that the *red part* or the *blue part* has been cancelled.

The merging complexity must be updated: the blue list is of size $2^{\mathcal{D}_b+\mathcal{A}_r}$ and the red list of size $2^{\mathcal{D}_r+\mathcal{A}_b}$, since both depend on the corresponding cancellation

Algorithm 1 MITM attack algorithm, following [19,20].

```
    Input: target hash value $(H_1, H_2)$, configuration of the attack (red bits, blue bits, cancellation points, matching points, $N$ bit-constraints on the outer part)
    Output: three-block message $M_1, M_2, M_3$ leading to the image $H_1, H_2$
 1: Determine the constant $c$ and value $h_t$ at the matching points
 2: for $2^{65}$ pairs $M_1, M_2$ do
 3:     Compute the inner part after absorbing $M_1, M_2$
 4:     If it does not satisfy the $N$ constraints, continue
 5:     for All possible values $x_G$ for the gray bits ▨ of $M_3$ do
                                                            ▷ Merging starts
 6:         Initialize a hash table $U$
 7:         for All values $x_R$ for the red bits 🟥 of $M_3$ do
 8:             for All values $y_B$ for the blue cancellation points in the path do
 9:                 Compute $f(x_R, x_G, y_B)$ and $g'(x_R, x_G)$
10:                 Store $x_R$ in the table $U$ at index $(y_B, g'(x_R, x_G), f(x_R, x_G, y_B))$
11:             end for
12:         end for
                                                            ▷ Red list constructed
13:         for All values $x_B$ for the blue bits 🟦 of $M_3$ do
14:             for All values $y_R$ for the red cancellation points in the path do
15:                 Compute $g(x_B, x_G, y_R)$ and $f'(x_B, x_G)$
16:                 Fetch the data in $U$ at index $(f'(x_B, x_G), y_R, g(x_B, x_G, y_R) \oplus c \oplus h_t)$
                                ▷ By construction, the matching and cancellation points agree
17:                 for Any obtained $x_R$ do
18:                     Set $M_3 = (x_B, x_R, x_G)$
19:                     Recompute the entire hash of $M_1, M_2, M_3$
20:                     If it equals $H_1, H_2$, Return
21:                 end for
22:             end for
23:         end for
24:     end for
25: end for
```

points. However, since each cancellation point is also a matching point, the merged list size is unchanged. The time and memory complexities of the full attack are updated to:

$$\begin{cases} T = \max\left(2^{65+N}, 2^{128-\mathcal{D}_r+\mathcal{A}_r}, 2^{128-\mathcal{D}_b+\mathcal{A}_b}, 2^{128-\mathcal{D}_m}\right) \\ M = \min\left(2^{\mathcal{D}_r+\mathcal{A}_b}, 2^{\mathcal{D}_b+\mathcal{A}_r}\right) \quad . \end{cases} \tag{7}$$

The attack can be summarized as Algorithm 1.

The attacks of [19,20] do not use both blue and red cancellations. We have tried to authorize both but did not arrive at better results. In fact, it seems that the optimal strategies are unbalanced: one of the lists will have more degrees of freedom and no cancellations, and the other one will have less degrees of freedom and many cancellations. This gives two lists of approximately equal size.

3.3 Automatic Search Strategies

The *configuration* of a MITM preimage attack is given by the coloring pattern of bits, the choice of cancellation points and conditions on the outer part. The time and memory complexities are determined by the number of blue and red bits in M_3 (blue and red degree of freedom), and the number of matching and cancellation bits, as per Equation 7.

Qin et al. show that the problem of finding the best MITM preimage attack can be reduced to a *Mixed-Integer Linear Programming* (MILP) problem, i.e., optimization under linear inequalities using real, integer and Boolean variables. Roughly speaking, the configuration is modeled using Boolean variables and the dominating term of the time and memory complexities (in $\log_2$) is optimized:

$$\begin{cases} T = \max\left(65 + N, 128 - \mathcal{D}_r + \mathcal{A}_r, 128 - \mathcal{D}_b + \mathcal{A}_b, 128 - \mathcal{D}_m\right) \\ M = \min\left(\mathcal{D}_r + \mathcal{A}_b, \mathcal{D}_b + \mathcal{A}_r\right) \quad . \end{cases} \tag{8}$$

Due to the number of colors to represent, the resulting MILP model requires 320×3 Boolean variables for each intermediate state, i.e., at least 320×6 per round. In this situation, the model is too large for the MILP solver to achieve optimal results. Instead, one will leave it running indefinitely, stop after some time and hope that the current result is close to optimal. This is why the 3-round attack on Ascon-Hash of [19] could be improved in [20] with a heuristic method that fixed part of the configuration to accelerate the search of solutions.

4 Using SAT to Improve the MITM Attacks

In this section, we explain our SAT modeling of the MITM attack. Its representation of the configuration is quite close to the MILP modeling, but the objective function is different (since we need to solve a satisfiability problem, not an optimization one).

4.1 Variables

A bit in the path can have the following colors: • blue 🟦; • red 🟥; • gray ⬛; • green 🟩, which means a linear combination of a blue and a red component (🟩 = 🟦+🟥); • white □, which is a nonlinear combination of blue and red, which cannot be used. Like [19,20], the color of each bit x_i is determined by three Boolean variables (x_i^b, x_i^r, x_i^w); however our correspondence is different. The variable x_i^b indicates whether the bit has a "blue part", x_i^r whether it has a "red part", x_i^w whether it is white (x_i^w dominates). The correspondence between variables and colors is given in Table 1.

In addition, for each bit in the path, Boolean variables indicate if the red or blue component are cancelled at this bit. Because they ultimately serve to avoid nonlinear effects, cancellations occur only in the state before p_S. We denote $A^{(i)}$ and $S^{(i)}$ as two intermediate states of round i, with the convention that

Table 1: Correspondence between Boolean variables and colors for a single bit.

x_i^b	x_i^r	x_i^w	Color	x_i^b	x_i^r	x_i^w	Color
0	0	0	Gray	1	0	0	Blue
0	0	1	White	1	0	1	White
0	1	0	Red	1	1	0	Green
0	1	1	White	1	1	1	White

$p_S \circ p_C$ corresponds to the transition from $A^{(i)}$ to $S^{(i)}$ and p_L corresponds to the transition from $S^{(i)}$ to $A^{(i+1)}$. For 3 rounds, this gives:

$$A^{(0)} \xrightarrow[p_S \circ p_C]{} S^{(0)} \xrightarrow[p_L]{} A^{(1)} \xrightarrow[p_S \circ p_C]{} S^{(1)} \xrightarrow[p_L]{} A^{(2)} \xrightarrow[p_S \circ p_C]{} S^{(2)} \xrightarrow[p_L]{} A^{(3)}$$

Next, we reuse two optimizations from Qin et al.:

- The state $A^{(0)}$ can be omitted. Indeed, consider a column y in $A^{(0)}$ and the corresponding column y' in $S^{(0)}$, we have:

$$y'_1 = y_4 \oplus y_3 y_2 \oplus y_3 y_1 \oplus y_3 \oplus y_2 y_1 \oplus y_2 \oplus y_1 \oplus y_0 \ .$$

  Since the inner part is fixed, y_1, y_2, y_3, y_4 are all gray. Therefore y'_1 and y_0 have the same color. Instead of starting from the first line of $A^{(0)}$ and deducing the colors of all bits, we can therefore start from the second line of $S^{(0)}$.
- The last p_L can be omitted. Indeed, it applies on the rows, and the output block H_1 is extracted from the outer part of the state, which is the first row. Therefore we can remove p_L and seek to obtain $p_L^{-1}(H_1)$ instead of H_1 as the first half of the desired hash.

4.2 Propagation of Colors

The colors of all bits are deduced from the coloring scheme of M_3 (and therefore, the colors of line 1 in $S^{(0)}$), the cancellations, and the N initial constraints on the outer part.

Initial State. We start with the constraints on the outer part which determine the colors in $S^{(0)}$. The goal is to fix some bits or linear combinations of bits in order to obtain some gray bits after the first S-Box layer. Following [19], we can use one or two constraints for each column $(y'_0, \dots, y'_4)$. Therefore we introduce two Boolean variables p_1 and p_2 for each column.

- In all cases, y'_2 is gray (y_0 does not intervene in its expression)
- By paying one bit of constraint (p_1 true), we can make either y'_0 or y'_4 gray (but not both). Indeed, by the ANF of the S-Box and the fact that $y_1, \dots, y_4$ are gray, we have:

$$y'_0 = \blacksquare \oplus y_1 y_0 \oplus y_1 \oplus y_0, \quad y'_4 = \blacksquare \oplus y_1 y_0 \oplus y_1$$

We can force y_1 to be 1, which results in y'_0 to be gray (and y'_4 to be colored), or y_1 to be 0, which results in y'_4 to be gray (and y'_0 to be colored)

– By paying one bit of constraint (p_2 true), we can make y'_3 gray. Indeed, we have:

$$y'_3 = (y_4 \oplus y_3)y_0 \oplus \blacksquare \oplus y_0$$

We can force $(y_4 \oplus y_3) = 1$, which results in y'_3 to be gray. Otherwise it will have the same color as y'_1.

Therefore, the clauses which constrain the initial state $S^{(0)}$ can be summarized as follows, for each column x:

1: $\neg x_1^b \vee \neg x_1^r$ ▷ Cannot be red and blue
2: $p_1 \vee \neg x_1^b \vee x_3^b$ ▷ Cancel color in x_3 (blue version)
3: $p_1 \vee \neg x_1^r \vee x_3^r$ ▷ Cancel color in x_3 (red version)
4: $p_2 \vee x_0^b \vee \neg x_1^b$ ▷ p_2 can cancel color in x_0
5: $p_2 \vee x_4^b \vee \neg x_1^b$ ▷ p_2 can cancel color in x_4
6: $x_4^b \vee x_0^b \vee \neg x_1^b$ ▷ Either x_0 or x_4 need to be colored
7: $p_2 \vee x_0^r \vee \neg x_1^r$
8: $p_2 \vee x_4^r \vee \neg x_1^r$
9: $x_4^r \vee x_0^r \vee \neg x_1^r$

Propagation through p_L. Each bit after p_L is the XOR of three bits located at different columns. If x_1, x_2, x_3 are the previous bits, the color of $y = x_1 \oplus x_2 \oplus x_3$ is determined as follows:

– If one of the x_i is white, then y is white
– If one of the x_i is blue, then y is blue
– If one of the x_i is red, then y is red *unless* we cancel red at this position

Blue cancellations were not mentioned at all in [19], but we did not find any advantage from them (they also make the problem significantly harder to solve). Therefore we focus on red cancellations. We introduce a Boolean variable c^r indicating a red cancellation at this position. We have the following constraints:

1: $(x_1^w \vee x_2^w \vee x_3^w) \implies y^w$ ▷ Propagation of white
2: $(x_1^b \vee x_2^b \vee x_3^b) \implies y^b$ ▷ Propagation of blue
3: $(x_1^r \vee x_2^r \vee x_3^r) \implies (y^r \vee c^r)$ ▷ Propagation of red, unless cancellation

Propagation through p_S. The propagation through p_S is studied column-wise. As there are no cancellations here, we simply look at the expression of each output bit of the S-Box from the input bits:

– If one of the inputs is blue (resp. red, resp. white), the output is blue (resp. red, resp. white)
– For any pair x_i, x_i such that the term $x_i x_j$ appears in the ANF, if x_i is blue (resp. red) and x_j is red (resp. blue), then the output is white

It is here that white bits (resulting from an AND between red and blue expressions) start to appear and ultimately contaminate the entire state.

4.3 Objective

A SAT modeling does not support integer variables, and furthermore, it does not optimize. Therefore the implementation of the *objective function* differs from the MILP.

Recall that our goal is to optimize the time and memory complexities given as Equation 8:

$$\begin{cases} T = \max\left(65 + N, 128 - \mathcal{D}_r + \mathcal{A}_r, 128 - \mathcal{D}_b, 128 - \mathcal{D}_m\right) \\ M = \min\left(\mathcal{D}_r, \mathcal{D}_b + \mathcal{A}_r\right) \quad . \end{cases} \tag{9}$$

where we have taken $\mathcal{A}_b = 0$ (no cancellation of blue). Fortunately, all the variables of these formulas can be expressed as the sums of Boolean variables: N is the sum of all (p_1, p_2) that determine the amount of constraints on the initial state, $\mathcal{D}_r$ is the number of red bits in the initial state, $\mathcal{D}_b$ the number of blue bits, $\mathcal{A}_r$ the number of cancellation variables set to "true", $\mathcal{D}_m$ the number of matching bits, i.e., the number of bits which are not white in line 0 in the last state.

Constraints on a number of Boolean variables which can be set to "true" are known as *cardinality constraints*, and a number of implementations are available in state-of-the-art SAT solvers. A k-cardinality constraint translates into a set of clauses an inequality of the form:

$$\sum_{i=1}^{n} x_i \leq k$$

where x_i are Boolean variables and k is a constant.

Minimizing the Time. To simulate the minimization of the time, we manually impose an upper bound on T and let the solver find a solution (or declare the problem "unsat"). having fixed the value of T, Equation 8 can be transformed into the system of inequalities:

$$\begin{cases} N \leq T - 65 \\ -\mathcal{D}_b + \mathcal{A}_b \leq T - 128 \\ -\mathcal{D}_r + \mathcal{A}_r \leq T - 128 \\ -\mathcal{D}_m \leq T - 128 \end{cases}$$

Next, we eliminate subtractions from the inequalities above by replacing variables with their negations. For example, we let $\overline{\mathcal{D}_b}$ be the sum of the negations of all Boolean variables x_i^b over the 64 controlled bits x_i of the initial state. We have $-\mathcal{D}_b = \overline{\mathcal{D}_b} - 64$. Finally, the obtained constraints are:

$$\begin{cases} N \leq T - 65 \\ \overline{\mathcal{D}_b} + \mathcal{A}_b \leq T - 64 \\ \overline{\mathcal{D}_r} + \mathcal{A}_r \leq T - 64 \\ \overline{\mathcal{D}_m} \leq T - 64 \end{cases}$$

4.4 Optimizations

In order to reduce the solving time down to a couple of minutes, we included the following optimizations.

Removing the Last S-Box. In practice, we are not interested in the entire state at the last round, but only in the outer part (i.e., the first line). The expression of this bit y'_0 after the final S-Box is the following:

$$\begin{aligned} y'_0 &= y_4y_1 \oplus y_3 \oplus y_2y_1 \oplus y_2 \oplus y_1y_0 \oplus y_1 \oplus y_0 \\ &= y_1\left(y_4 \oplus y_2 \oplus y_0 \oplus 1\right) \oplus y_3 \oplus y_2 \oplus y_0 \ . \end{aligned}$$

If we want this bit not to be white, we can cancel the red part of y_1 and the red part of $y_4 \oplus y_2 \oplus y_0 \oplus 1$ (blue cancellations are not allowed according to the previous point).

This technique has several advantages: first, we no longer need to consider the application of the last S-Box. Detection of matching bits is done by testing whether a column at the input of the last S-Box is completely non-white. Furthermore, we will only need to pay for two cancellations per matching bit in the worst case, as opposed to 4 if we did not perform this refined analysis. In practice, we observed that this technique simplifies the search for the 4-rounds attack. It is not required in other cases.

Symmetry. We noticed that all solutions generated by the solver for Ascon reduced to 4 rounds were circularly symmetric, meaning that the diffusion pattern repeated 32 positions later. Concerning the reduction to 3 rounds, the solutions also exhibited some symmetry.

We believe that the prevalence of symmetries in the solutions found is due to the heuristics of the automated tools used (SAT, MILP). However, such circularly symmetric solutions exist only because the design of Ascon allows for their existence. Ascon consists of two main parts: $p_S \circ p_C$, which act locally on each column, and p_L, which acts laterally by mixing the rows. By construction of the model, the solver seeks to produce as few green bits as possible after the application of p_L, as green bits have a high probability of becoming white after passing through the S-box. Thus, the presence of p_L encourages the model to have the same color at relative positions: before the application of p_L, if the model decides that a bit will be of a given color, it will propagate this color to relative positions on the same row, creating repeated patterns. Since the offsets of this repetition are uniform along a given line, we expect the width of a pattern to divide the total length of the line (64).

Surely, there are non-symmetric solutions, as nothing in Ascon forces the appearance of this symmetry. However, it seems that restricting to symmetric solutions still allows to find the best complexities. It reduces significantly the number of variables and clauses.

Solver Choice. We compared several solvers available through the Python library PySAT [16]. The best results (fastest solving time) were obtained with *Glucose4* combined with the cardinality constraints encoded by *kmtotalizer*. We kept this combination afterwards.

4.5 Results

Improved Solutions. The symmetric model runs in the order of minutes on a desktop computer to find both 3- and 4-round attacks. While we did not improve the time complexities reported in [19] for 4-round Ascon (around 2^{124}, using 4 bits of matching) and in [20] for 3-round Ascon (around 2^{114}, using 14 bits of matching), we could reduce the memory complexity using the following parameters:

- 3 rounds: $\mathcal{D}_b = 14$, $\mathcal{D}_r = 24$, $\mathcal{D}_m = 14$, $\mathcal{A}_r = 10$ (Figure 3)
- 4 rounds: $\mathcal{D}_b = 4$, $\mathcal{D}_r = 34$, $\mathcal{D}_m = 4$, $\mathcal{A}_r = 30$ (Figure 4)

which give respective memory complexities of 2^{24} and 2^{34}.

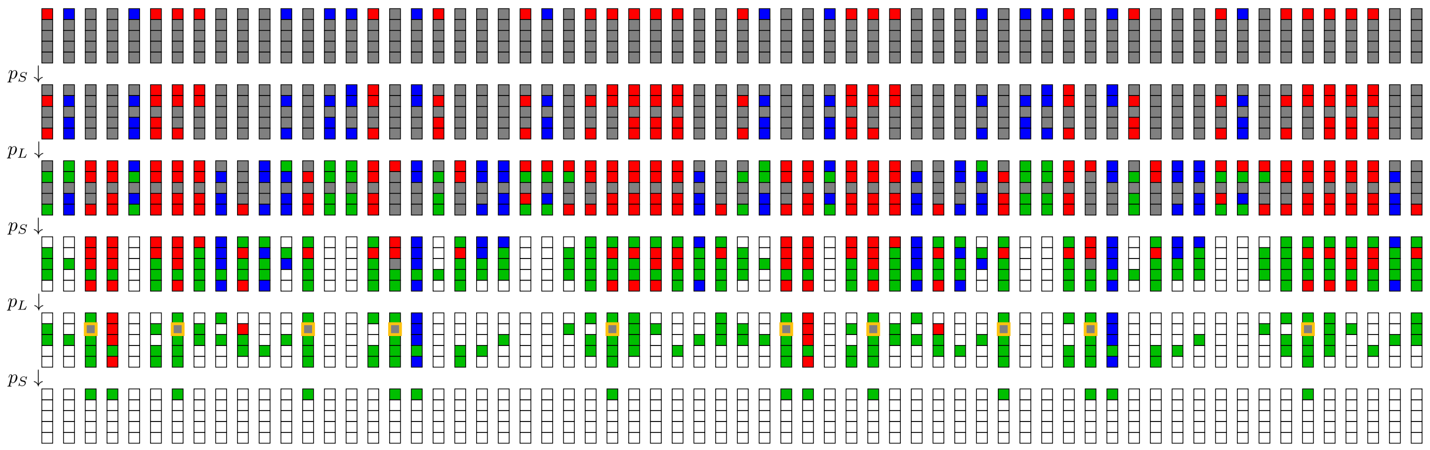
Optimality of Results. By running the SAT solver with stronger constraints, we can determine whether the problem has solutions. In particular, the formula for the time complexity (Equation 8) shows that to achieve a complexity $2^{128-\alpha}$, we need $\mathcal{D}_m \geq \alpha$. Therefore, the strategy to prove the optimality of a time complexity is to impose a cardinality constraint on the matching bits. To speed up the search, we can note that the diffusion pattern of Ascon is invariant under circular permutation; therefore we can always impose a matching bit at a fixed position.

For Ascon with 3 rounds, the SAT solver did not manage to conclude, i.e., to prove unsatisfiability or find a solution. This might be because the number of matching bits for 3 rounds is quite large compared to the other cases.

For Ascon with 4 rounds, we search for a valid configuration with $\mathcal{D}_m \geq 5$. By symmetry, we first impose bit 0 as a matching bit. Then, we cut the problem into 16 easier SAT problems by imposing a second matching bit at position 1, then 2, and so on (by circular permutation invariance of Ascon and symmetry, it is not necessary to check pairs beyond 16). The entire computation takes about 3 hours. This proves that there is no better 4-round MITM attack of this type, assuming symmetry and cancellation of only one color. Finally, fixing the time complexity of the 4-round attack at 2^{124}, we find that it is impossible to increase the number of gray bits further, proving that our memory complexity is also optimal.

For Ascon with 5 rounds, we search for a valid configuration with $\mathcal{D}_m \geq 1$. Imposing that bit 0 is a matching bit leads the SAT solver to conclude unsatisfiability after a few minutes. This proves that there is no 5-round MITM attack of this type, assuming symmetry and cancellation of only one color. Our results are summarized in Table 2.

The optimality, indicated by $\star$, is subject to the aforementioned two constraints and is for this Meet-in-the-Middle attack.



3-round attack found with our SAT-based approach, improving the one of [20].

Fig. 3: Cancellations of red bits are represented by yellow squares. There are 44 bits of additional constraints which are imposed on the inner part of the first state to guarantee the first transition through p_S.

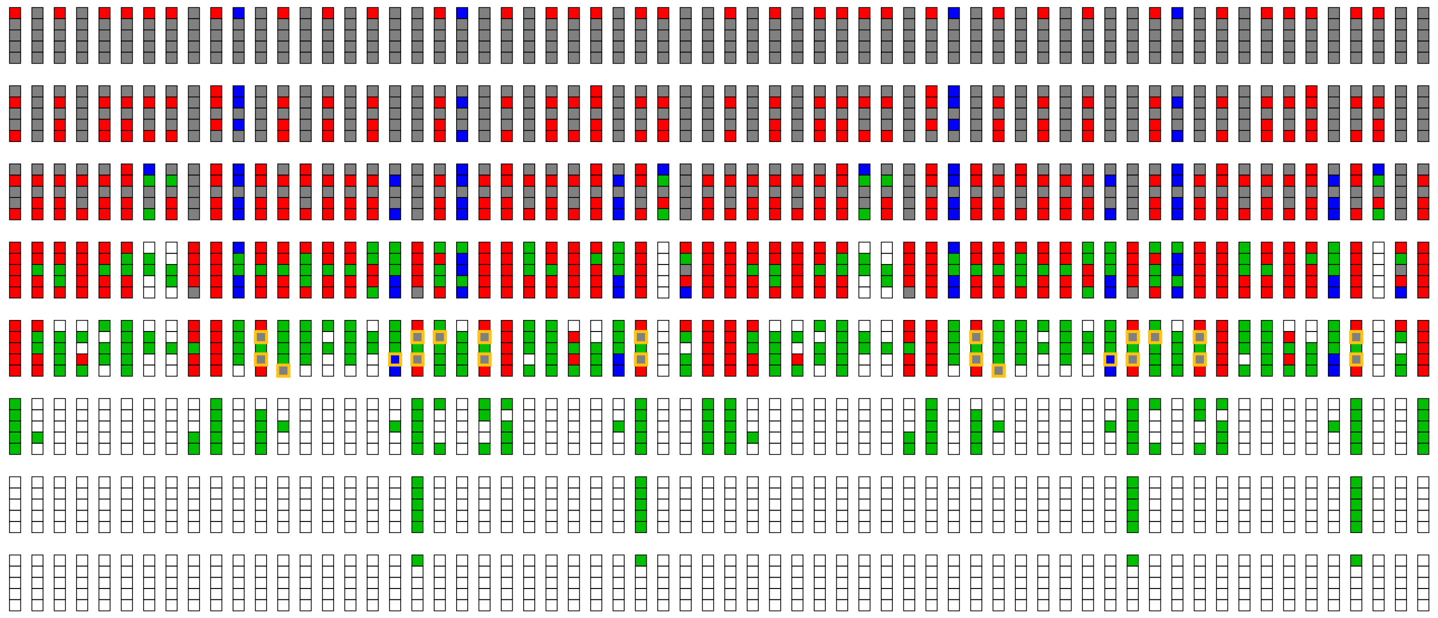


Fig. 4: Optimal 4-round attack found with our SAT-based approach, improving the one of [20]. Cancellations of red bits are represented by yellow squares. There are 52 bits of additional constraints which are imposed on the inner part of the first state to guarantee the first transition through p_S. The first 26 are the following, where $A_{i,j}$ is the bit at line j and column i in the first state:

$$A_{0,3} \oplus A_{0,4} = 1,\ A_{0,1} = 1,\ A_{2,1} = 1,\ A_{4,1} = 1,\ A_{5,1} = 1,\ A_{6,3} \oplus A_{6,4} = 1,\ A_{6,1} = 1,\ A_{7,3} \oplus A_{7,4} = 1,\ A_{9,1} = 0,$$
$$A_{10,1} = 0,\ A_{12,1} = A_{14,1} = A_{16,1} = A_{19,1} = 1,\ A_{20,3} \oplus A_{20,4} = 1,\ A_{20,1} = 1,\ A_{22,3} \oplus A_{22,4} = 1,$$
$$A_{22,1} = 1,\ A_{24,1} = 1,\ A_{25,3} \oplus A_{25,4} = 1,\ A_{25,1} = 1,\ A_{28,3} \oplus A_{28,4} = 1,\ A_{28,1} = 1,\ A_{29,1} = 1$$

Table 2: Summary of results obtained. $\star$: optimal among MITM attacks of this form, under the symmetry constraint. $\star\star$: optimal when the time complexity is at 2^{124}. We consider only the dominating term in the complexity.

Number of Rounds	Authors	Time Complexity	Memory Complexity
3 rounds	[20]	2^{114}	2^{30}
	Ours	2^{114}	2^{24}
4 rounds	[19]	2^{124}	2^{54}
	Ours	2^{124} ($\star$)	2^{34} ($\star\star$)
5 rounds	**Ours**	2^{128} ($\star$)	—

While we did not improve the time complexities reported in [19] for 4-round Ascon (around 2^{124}, using 4 bits of matching) and in [20] for 3-round Ascon (around 2^{114}, using 14 bits of matching), we reduced the memory complexities as seen in Table 2. The path for the 4-round attack is shown in Figure 4, with $\mathcal{D}_b = 4$, $\mathcal{D}_r = 34$, $\mathcal{D}_m = 4$, $\mathcal{A}_r = 30$.

5 New MILP Model for Differential Bounds

Our second simple model aims at obtaining lower bounds on the probability of differential characteristics for the Ascon permutation. The bounds proven so far are only tight up to 3 rounds despite years of investigation as summarized in Table 3, with different methods such as MILP [18], SAT and SMT [11]. The state of the art for lower bounds is given by a tree extension model from [15].

Table 3: Currently known differential bounds of the Ascon permutation restricted to R rounds.

	Upper bound			Lower Bound		
R	Bound	Method	Ref.	Bound	Method	Ref.
1	2^{-2}	DDT		2^{-2}	DDT	
2	2^{-8}	DDT + β		2^{-8}	DDT + β	
3	2^{-40}	ndltool	[7]	2^{-40}	MILP	[18]
4	2^{-107}	ndltool		2^{-86}	Tree extension	
5	2^{-190}	CP	[7],[12]	2^{-100}	Tree extension	
6	2^{-305}	CP	[12]	2^{-129}	Tree extension	
7				2^{-131}	Tree extension	
8				2^{-172}	Tree extension	[15]
9				2^{-186}	Tree extension	
10				2^{-215}	Tree extension	
11				2^{-229}	Tree extension	
12				2^{-258}	Tree extension	

One interesting remark is that many of the lower bounds on the probability are computed from the lower bounds of a lower number of rounds. For example, the current lower bound on 12 rounds is computed from the bound on 4 rounds: $2^{-258} = 2^{-86\times 3}$. As such, improving the lower bounds give us immediate results for higher rounds. This also means that the results for higher rounds are most likely far to be tight, which is especially visible for the bound on 7 rounds: $2^{-131} = 2^{-129} \times 2^{-2}$.

The method used to calculate these bounds varies notably, with SAT and SMT models giving great results [11], before being outclassed by a tree extension model [15] that is the current state of the art for lower bounds. We will focus here on the MILP models, an approach used by [18] to close the gap for 3 rounds.

Our main objective was to reduce the gap between the lower and upper bounds. For now, the bounds for 4 rounds are **[86, 107]** (or to be more exact, since the bounds are probabilities, $[2^{-107}, 2^{-86}]$). However, [18] showed that there exists a 4 round trail with 43 active S-boxes. This mean that, if we want to reduce the gap by improving the lower bound on the weight, we cannot take the number of active S-boxes as objective, since we know that the minimum number of active S-boxes will be ≤ 43 and as such we will not be able to tell a better precision than $43 \times 2 = 86$ but have to model the transition weights.

5.1 A new MILP Approach: using the Hamming Weight

Our idea to improve the existing bounds is to consider a lossy model, as it was done in [5] regarding division property related problems. This would allow us to get calculations done much quicker, but at the cost of getting less accurate results. The main difficulty is to find a right balance between accuracy and time complexity. As such, we tried to model the internal state of Ascon using the Hamming weight of columns.

The Hamming weight of a column is defined by the total number of active bits in this column. As such, instead of modeling 320 bits, we can manipulate 64 integers between 0 and 5. This approach works particularly well with MILP models, as they deal natively with integer variables.

5.2 Modeling p_S

To model the non linear layer, we need a way to represent the weight of each transition through the Ascon S-Box. Each transition has a weight of 2, 3, 4 or 0 (for the trivial $0 \to 0$ differential transition), and thus for each of them we create binary variables w_2, w_3, w_4 and w_0 with the constraint that $w_2[i] + w_3[i] + w_4[i] + w_0[i] \ = \ 1$.

Now that we have a way to represent the weights of the transition, we then have to find a way to model the DDT, then link both of them together. The classical way of modelling the DDT into inequalities is to use the convex hull operator. This operator takes a cloud of points in n dimensions and returns the convex hull associated with this cloud of points, which can take the form of inequalities, which is what we want. In our case, we want to know what is

Table 4: hwDDT of ASCON's S-box: maximum number of solutions as a function of the Hamming weight of the input and output.

hw	0	1	2	3	4	5
0	32	0	0	0	0	0
1	0	0	8	8	8	4
2	0	8	8	8	8	4
3	0	8	8	4	4	4
4	0	4	4	4	4	2
5	0	4	4	4	0	0

the minimum weight (the worst case) of a differential transition where the input difference has a Hamming weight of $HW(i)$ and the output has a Hamming weight of $HW(j)$. As such, our $HWDDT$ can be expressed $(i, j) \in \{0, ..., 5\}$ as:

$$hwDDT(i, j) = max(DDT(a, b) \mid hw(a) = i, hw(b) = j),$$

and is given in Table 4.

More precisely, this new DDT contains the best transitions between an input and an output of given Hamming weight. Hopefully, it is quite straightforward to describe a transition $a \to b$ through this DDT:

$$\begin{array}{ccc} a \geq 2w_4 & & \\ a \leq 5 - w_4 & a \leq 5 - 5w_0 & a + b \geq 3w_2 \\ b \geq w_4 & b \leq 5 - 5w_0 & a \geq w_2 \\ a \geq w_3 & b \leq 5 - w_2 & b \geq w_2 \\ b \geq w_3 & a \leq 5 - 2w_2 & 2a + b \leq 15 - 7w_2 \end{array}$$

Note that for the first and last non-linear layers, the constraints are simpler since we can assume that the best input or output will be selected. Furthermore, we only model ASCON from the output of the first non-linear layer to the input of the last one.

5.3 Modeling p_L

To model the linear layer, we need to express the relation between the bits of the state before and after this step. If we denote respectively by y and x these states, we have for all $(i, j) \in \{0, ..., 5\} \times \{0, ..., 64\}$:

$$x[i][j] = y[i][j] \oplus y[i][(j - d_1) \bmod 64] \oplus y[i][(j - d_2) \bmod 64],$$

where d_1 and d_2 depend on the row index. In [18], the authors proposed to use an extra binary variable per state bit to model the equation in a MILP-compliant form:

$$y[i][j] + y[i][(j - d_1) \bmod 64] + y[i][(j - d_2) \bmod 64] + x[i][j] = 2z[i][j].$$

However this modeling seems to be quite inefficient, making the model very slow to solve. Our idea is to describe as accurately as possible the possible transitions through the linear layer without going down to the bit level. To do so we introduce the variables x_{row}, x_{col}, y_{row} and y_{col} corresponding to the Hamming weight of the rows and columns of both states x and y (note that x_{col} and y_{col} are not new since they respectively correspond to the Hamming weight of the input and output of the S-boxes which are used in the modelization of the non-linear part).

First there are many straightforward relations between those states. For instance, $\sum_{i=0}^{4} x_{row}[i] = \sum_{i=0}^{6} 3x_{col}[i]$ and the same equality holds for y. It is also easy to add a constraint ensuring that an active column of y should at least activate the same column on x or one of the 10 associated columns of y. We also add the following constraint on rows:

$$3y_{row}[i] = x_{row}[i] + 2z,$$

where z is an extra integer variable, representing the number of cancellations occurring on the row.

5.4 Callbacks

Our model is very fast compared to previous ones, and in particular compared to [18]. However, the results are far from being accurate as many *false* trails are solutions of the model. To strengthen the model we use the callback functionality of the Gurobi MILP solver [14]. It allows to add an extra verification each time a solution of the model is found. First, for each linear layer, we check whether the pattern of active columns is possible using Gaussian elimination as it was done in [4]. If not we add an extra constraint to remove the pattern and the model continues to search for another solution. Finally, the whole trail is checked using an exact model. Note that the inequalities added to the model during the callback only involve the weight of the transitions as they all are binary variables.

5.5 Results

Our model is fast enough to retrieve the lower bound of the weight of differential characteristics up to 3 rounds. We also improve the lower bound for 7-round, showing that the minimal weight is at least 135 while the previous bound was 131. However, note that we included into the model the known differential bounds for rounds from 1 to 6, as it improves a lot the solving process. All these results are obtained on a regular desktop computer using at most days of computation.

6 Conclusion

In this work we proposed several improvements for the modelization of important cryptanalysis problems related to the security of Ascon. We successfully decrease the running times required to search for some instances of both Meet-in-the-middle preimage attacks on Ascon-Hash and lower bounds on the weight of differential characteristics on Ascon inner permutation, and obtained new results as well. The techniques we described show that it is sometimes more efficient to rely on simple modelizations, even though they are not exact, and we believe they could be used to improve models dedicated to other primitives.

Acknowledgments. This work has been partially supported by the French Agence Nationale de la Recherche through the OREO project under Contract ANR-22-CE39-0015, and through the France 2030 program under grant agreement No. ANR-22-PECY-0010.

References

1. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Selected Areas in Cryptography. LNCS, vol. 5381, pp. 103–119. Springer
2. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the sponge: Single-pass authenticated encryption and other applications. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7118, pp. 320–337. Springer (2011)
3. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Cryptographic sponge functions (2011), https://keccak.team/files/CSF-0.1.pdf
4. Boura, C., Derbez, P., Funk, M.: Related-key differential analysis of the AES. IACR Trans. Symmetric Cryptol. **2023**(4), 215–243 (2023). https://doi.org/10.46586/TOSC.V2023.I4.215-243
5. Derbez, P., Lambin, B.: Fast MILP models for division property. IACR Trans. Symmetric Cryptol. **2022**(2), 289–321 (2022). https://doi.org/10.46586/TOSC.V2022.I2.289-321
6. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer **10**(6), 74–84 (1977)
7. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Cryptanalysis of ascon. In: Nyberg, K. (ed.) Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9048, pp. 371–387. Springer (2015). https://doi.org/10.1007/978-3-319-16715-2_20
8. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: Lightweight authenticated encryption and hashing. J. Cryptol. **34**(3), 33 (2021). https://doi.org/10.1007/s00145-021-09398-9
9. Eichlseder, M.: TikZ libraries for crypto, https://extgit.iaik.tugraz.at/meichlseder/tikz
10. Erlacher, J., Mendel, F., Eichlseder, M.: Bounds for the security of ascon against differential and linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2022**(1), 64–87 (2022). https://doi.org/10.46586/TOSC.V2022.I1.64-87
11. Erlacher, J., Mendel, F., Eichlseder, M.: Bounds for the security of ascon against differential and linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2022**(1), 64–87 (2022). https://doi.org/10.46586/TOSC.V2022.I1.64-87
12. Gérault, D., Peyrin, T., Tan, Q.Q.: Exploring differential-based distinguishers and forgeries for ASCON. IACR Trans. Symmetric Cryptol. **2021**(3), 102–136 (2021). https://doi.org/10.46586/TOSC.V2021.I3.102-136

13. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced meet-in-the-middle preimage attacks: First results on full tiger, and improved results on MD4 and SHA-2. In: ASIACRYPT. LNCS, vol. 6477, pp. 56–75. Springer (2010). https://doi.org/10.1007/978-3-642-17373-8_4
14. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023), https://www.gurobi.com
15. Hirch, S.E., Mella, S., Mehrdad, A., Daemen, J.: Improved differential and linear trail bounds for ASCON. IACR Trans. Symmetric Cryptol. **2022**(4), 145–178 (2022). https://doi.org/10.46586/TOSC.V2022.I4.145-178
16. Ignatiev, A., Morgado, A., Marques-Silva, J.: PySAT: A Python toolkit for prototyping with SAT oracles. In: SAT. pp. 428–437 (2018). https://doi.org/10.1007/978-3-319-94144-8_26, https://doi.org/10.1007/978-3-319-94144-8_26
17. Li, H., He, L., Chen, S., Guo, J., Qiu, W.: Automatic preimage attack framework on ascon using a linearize-and-guess approach. IACR Transactions on Symmetric Cryptology **2023**(3), 74–100 (Sep 2023). https://doi.org/10.46586/tosc.v2023.i3.74-100, https://tosc.iacr.org/index.php/ToSC/article/view/11185
18. Makarim, R.H., Rohit, R.: Towards tight differential bounds of ascon A hybrid usage of SMT and MILP. IACR Trans. Symmetric Cryptol. **2022**(3), 303–340 (2022). https://doi.org/10.46586/TOSC.V2022.I3.303-340
19. Qin, L., Hua, J., Dong, X., Yan, H., Wang, X.: Meet-in-the-middle preimage attacks on sponge-based hashing. In: EUROCRYPT (4). Lecture Notes in Computer Science, vol. 14007, pp. 158–188. Springer (2023). https://doi.org/10.1007/978-3-031-30634-1_6
20. Qin, L., Zhao, B., Hua, J., Dong, X., Wang, X.: Weak-diffusion structure: Meet-in-the-middle attacks on sponge-based hashing revisited. IACR Cryptol. ePrint Arch. p. 518 (2023), https://eprint.iacr.org/2023/518
21. Sun, L., Wang, W., Wang, M.: Accelerating the search of differential and linear characteristics with the sat method. IACR Transactions on Symmetric Cryptology **2021**(1), 269–315 (Mar 2021). https://doi.org/10.46586/tosc.v2021.i1.269-315, https://tosc.iacr.org/index.php/ToSC/article/view/8840
22. Turan, M.S., McKay, K., Chang, D., Kang, J., Waller, N., Kelsey, J.M., Bassham, L.E., Hong, D.: Status report on the final round of the nist lightweight cryptography standardization process (2023)