IACR Transactions on Symmetric Cryptology
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–35. DOI:XXXXXXXX

Exploring Differential-Based Distinguishers and Forgeries for ASCON

David Gerault[1,2], Thomas Peyrin[1] and Quan Quan Tan[1]

[1] Nanyang Technological University, Singapore
[2] University of Surrey, UK

dagerault@gmail.com,thomas.peyrin@ntu.edu.sg,quanquan001@e.ntu.edu.sg

Abstract. Automated methods have become crucial components when searching for distinguishers against symmetric-key cryptographic primitives. While MILP and SAT solvers are among the most popular tools to model ciphers and perform cryptanalysis, other methods with different performance profiles are appearing. In this article, we explore the use of Constraint Programming (CP) for differential cryptanalysis on the Ascon authenticated encryption family (first choice of the CAESAR lightweight applications portfolio and current finalist of the NIST LWC competition) and its internal permutation. We first present a search methodology for finding differential characteristics for Ascon with CP, which can easily find the best differential characteristics already reported by the Ascon designers. This shows the capability of CP in generating easily good differential results compared to dedicated search heuristics. Based on our tool, we also parametrize the search strategies in CP to generate other differential characteristics with the goal of forming limited-birthday distinguishers for 4, 5, 6 and 7 rounds and rectangle attacks for 4 and 5 rounds of the Ascon internal permutation. We propose a categorization of the distinguishers into black-box and non-black-box to better differentiate them as they are often useful in different contexts. We also obtained limited-birthday distinguishers which represent currently the best known distinguishers for 4, 5 and 6 rounds under the category of non-black-box distinguishers. Leveraging again our tool, we have generated forgery attacks against both reduced-rounds Ascon-128 and Ascon-128a, improving over the best reported results at the time of writing. Finally, using the best differential characteristic we have found for 2 rounds, we could also improve a recent attack on round-reduced Ascon-Hash.

Keywords: Differential Cryptanalysis · ASCON · Constraint Programming · Rectangle Attacks · Limited-birthday · Forgery

1 Introduction

With the increasing need for a cryptographic primitive providing both encryption and authentication, so-called authenticated encryption (AE), as well as the rise of lightweight cryptography (cryptography for devices with important constraints with regards to area, energy/power consumption, latency, etc.), the National Institute of Standards and Technology (NIST) decided to start a new cryptographic competition in 2019 for lightweight authenticated encryption [oST21]. This standardization effort saw the submission of 57 candidates and Ascon [DEMS21b] (also first choice for the "lightweight applications" final portfolio of the CAESAR competition [CAE19]) is currently one of the finalists of the competition. As a potential primitive to be used as a standard and in order to provide a good comparison of the relative security provided by the finalists, it is important to have a wide range of cryptanalysis conducted.

As more cipher designs along with different block cipher modes are being published, cryptanalysts' playing field could only increase, giving them more areas to explore and probe. Differential cryptanalysis [BS90, BS91], one of the oldest forms of statistical cryptanalysis in modern cryptography, has developed and spawned many other more advanced variants such as higher-order differential cryptanalysis [Lai94] and boomerang attacks [Wag99]. The set of distinguishing properties is also getting more diverse, with attackers scrutinizing for example zero-sum [AM, BCC11], subspace [LMS+15] or limited-birthday [GP10, IPS13, JNP13] properties.

As the playing field grew larger, the cryptanalysts' toolbox grew bigger too, dedicated heuristic search algorithms and automated tools such as MILP/MIP [MWGP11], SAT [MP13] and CP [GMS16] are being employed to find differential/linear characteristics as well as other patterns in cryptographic primitives. Recently, machine learning and in particular neural networks seem to show some ability in finding statistical patterns in cipher queries [Goh19]. While one can count a very large number of works using MILP in cryptanalysis, articles employing CP for heuristic methods remain relatively scarce. With CP focusing on solving combinatorial problems using logical inferences, it deserves perhaps more attention in cryptanalytic applications.

Our contributions. In this article, we focus our attention on the differential cryptanalysis of the Ascon permutation. We propose four contributions.

First, we present a methodology that uses CP to automatically find good differential characteristics for Ascon. We show that we can replicate the designers' results in [DEMS19], which were produced with a complex dedicated heuristic algorithm. The advantage of using CP not only allows us to efficiently model the permutation, it also taps into the rich number of search strategies provided by different solvers. Our method is quite generic and can be applied to other ciphers with minor tweaking. Other than having the ability to choose from a wide variety of solvers, another advantage is that it can be easily parameterized to find differential characteristics with specific properties that we might want to enforce. For example, in our case, we use CP to find specific differential characteristics to be used in limited-birthday distinguishers.

Secondly, for better categorization, we propose to split the distinguishers into two types: black-box and non-black-box distinguishers which separates the distinguishers into those that can attack keyed permutations and those that can attack unkeyed permutations. We then use the differential characteristics found by CP to construct black-box/non-black-box limited-birthday distinguishers and rectangle distinguishers for the Ascon permutation. We notably present distinguishers for the Ascon permutation reduced to 4, 5, 6 and 7 rounds. All of our results are summarized in Table 1.

Thirdly, again using a specific parametrization of our CP tool, we find differential characteristics that allow us to obtain improvements for forgery attacks on reduced Ascon-128 and Ascon-128a compared to the state-of-the-art (see Table 1).

Lastly, using a similar strategy as in [ZDW19] and a new differential characteristic with a higher probability, we could improve the attacks on Ascon-Hash (Ascon-Hasha) reduced to 2 rounds (see Table 1).

Our work does not threaten the security of Ascon but gives a better understanding of the resistance of this candidate with regards to differential cryptanalysis-based attacks.

Table 1: Summary of results on permutation and attacks on reduced-round Ascon-128, Ascon-128a and Ascon-Hash (Ascon-Hasha). Complexities are expressed in number of primitive calls. BB and NBB represent black-box and non-black-box distinguishers. Note that Ascon-128, Ascon-128a, Ascon-Hash and Ascon-Hasha use 6, 8, 12 and 8 rounds of permutation in the iteration phase respectively. All of the primitives use 12 rounds for the initialization and finalization (if applicable). For generic complexity on zero-sum distinguishers, refer to Section 4.5 for more detail.

Distinguishers

# rounds	Type	BB	NBB	Comp. ($\log_2$)	Generic Comp. ($\log_2$)	Ref.
4	Diff.-Linear	✓		2	—	[DEMS15]
	Rectangle	✓		15.57	320	This paper
	Integral	✓		5	—	[RHSS21]
	Differential	✓		108	320	[DEMS21b]
	Linear	✓		101	320	[DEMS15]
	Limited-birthday	✓		8	≥ 204.81	This paper
	Rectangle		✓	9.57	320	This paper
	Zero-sum		✓	5	9.41	[DEMS15]
	Limited-birthday		✓	1	119	This paper
5	Rectangle	✓		79.57	320	This paper
	Integral	✓		16	—	[RHSS21]
	Differential	✓		191	320	[DEMS21b]
	Linear	✓		189	320	[DEMS15]
	Limited-birthday	✓		65	≥ 160.23	This paper
	Truncated Diff.	✓		108	—	[Tez16]
	Rectangle		✓	44.57	320	This paper
	Zero-sum		✓	10	10.7	[DEMS15]
	Limited-birthday		✓	1	75.01	This paper
6	Integral	✓		31	—	[RHSS21]
	Zero-sum		✓	10	10.7	[DEMS15]
	Limited-birthday		✓	2	≥ 15.18	This paper
7	Integral	✓		60	—	[RHSS21]
	Zero-sum		✓	18	18.00	[Tod15]
	Limited-birthday		✓	34	≥ 37.14	This paper

Forgeries

Primitive	# Rounds	Target	Scenario	Type	Complexity ($\log_2$)	Ref.
Ascon-128	3/12	Finalization	Nonce-respecting	Differential	34	[DEMS15]
		Finalization	Nonce-respecting	Differential	32.76	This paper
	4/12	Finalization	Nonce-respecting	Differential	102	[DEMS15]
		Finalization	Nonce-respecting	Differential	96.61	This paper
		Finalization	Nonce-misuse	Cube tester	9	[LZWW17]
	5/12	Finalization	Nonce-misuse	Cube tester	17	[LZWW17]
	6/12	Finalization	Nonce-misuse	Cube tester	33	[LZWW17]
Ascon-128a	3/12	Iteration	Nonce-respecting	Differential	117	This paper
		Finalization	Nonce-respecting	Differential	20	This paper

Collision attacks

Primitive	# rounds	Complexity ($\log_2$)	Ref.
Ascon-Hash (Ascon-Hasha)	2/12 (2/8)	103	This paper
Ascon-Hash (Ascon-Hasha)	2/12 (2/8)	125	[ZDW19]

Outline. In Section 2, we briefly describe Ascon and some notations that will be used throughout the paper. In Section 3, we recall the first relevant works that inspired us to select CP as our preferred automated tool and the way to employ CP to find good differential characteristics. We then explain our choice of CP solver and how we model Ascon permutation to find differential characteristics. In Section 4, we explain how one can build limited-birthday distinguishers and rectangle distinguishers for the Ascon permutation using characteristics from our CP tools. In Section 5, we use our CP tool to generate special differential characteristics for forgery attacks on reduced-round Ascon-128 and Ascon-128a. In Section 6, we use a differential characteristic generated to improve the attack on reduced-round Ascon-Hash (Ascon-Hasha). Lastly, we draw conclusions and identify future works in Section 7.

2 Preliminaries

2.1 Differential cryptanalysis

Differential cryptanalysis was first proposed by Eli Biham and Adi Shamir in 1991 to tackle DES-like cryptosystem [BS90, BS91]. It is a form of chosen-plaintext attack where the attacker gets to query multiple pairs of plaintexts with a certain input difference and aims to observe a certain output difference. To measure whether an attack is feasible or more generally its complexity cost, one needs to evaluate the probability of the differential or the differential characteristics. Let $f : 2^m \to 2^n$ be a vectorial Boolean function from m bits to n bits. Then, the differential probability from an input difference Δ_{in} to an output difference Δ_{out} is given by

$$\mathbb{P}(\Delta_{in} \to \Delta_{out}) = \frac{\#\{x|f(x) \oplus f(x \oplus \Delta_{in}) = \Delta_{out}\}}{2^m}$$

In the case of a cryptographic primitive using a Substitution box, or Sbox, one important tool in differential cryptanalysis is the Difference Distribution Table (DDT). This helps to record the probabilities $\mathbb{P}(\Delta_{in} \to \Delta_{out})$ in a table format for all possible Δ_{in} and Δ_{out}. In the case of searching for differential characteristics, we can refer to the DDT to look for possible (good) differential transitions through the Sboxes.

2.2 A brief description of the Ascon family [DEMS21a]

Overview. The Ascon family of authenticated encryption schemes uses a sponge duplex construction with a key, k, of length 128 bits. In the ongoing NIST lightweight cryptography competition, the designers recommended two instances of the family, namely Ascon-128 and Ascon-128a. The main differences between the two are that Ascon-128 has 64-bit data block which is also known as the rate part, with $b = 6$ rounds of a permutation p (see Figure 1) whereas Ascon-128a has a 128-bit data block, with $b = 8$ rounds of permutation p. In both instances, the state and the tag sizes are 320 bits and 128 bits respectively. The number of permutation rounds for initialization and finalization is $a = 12$. In Figure 1, we show the encryption operation of the Ascon design.

Permutation. The permutation, p, has three sub-functions, namely: the addition of constants (p_C), the substitution layer (p_S) and the linear diffusion layer (p_L). Since we are focusing on differential cryptanalysis in this paper, the effects from p_C can be ignored for most parts. The internal state can be visualized as a (5×64)-bit array with the rate part at row 0 and rows 0 & 1 for Ascon-128 and Ascon-128a respectively. The function p_S applies 64 identical parallel 5-bit Sboxes on each column of the array. p_L

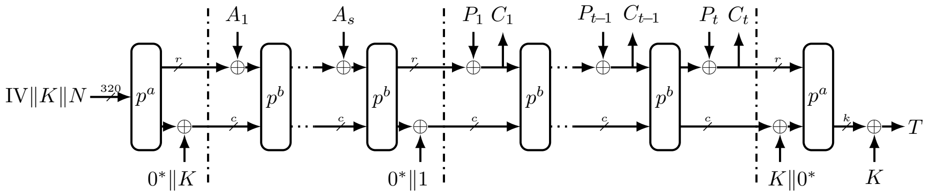

Figure 1: Encryption in Ascon.

occurs independently in each row; viewing the r^{th} row as a 64-bit word, w_i, each word being replaced by the XOR of three rotated values of the word:

$$\begin{aligned}
w_0 &\leftarrow w_0 \oplus (w_0 \ggg 19) \oplus (w_0 \ggg 28)\\
w_1 &\leftarrow w_1 \oplus (w_1 \ggg 61) \oplus (w_1 \ggg 39)\\
w_2 &\leftarrow w_2 \oplus (w_2 \ggg 1) \oplus (w_2 \ggg 6)\\
w_3 &\leftarrow w_3 \oplus (w_3 \ggg 10) \oplus (w_3 \ggg 17)\\
w_4 &\leftarrow w_4 \oplus (w_4 \ggg 7) \oplus (w_4 \ggg 41)
\end{aligned}$$

The Algebraic Normal Form (ANF) of the Sbox, is given by

$$\begin{aligned}
y_0 &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_1x_2 \oplus x_1x_4\\
y_1 &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3\\
y_2 &= x_1 \oplus x_2 \oplus x_4 \oplus x_3x_4 \oplus 1\\
y_3 &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_0x_3 \oplus x_0x_4\\
y_4 &= x_1 \oplus x_3 \oplus x_4 \oplus x_0x_1 \oplus x_1x_4
\end{aligned}$$

Note that in this paper, we will start the round count from 0. Thus, the "first" round refers to the 0^{th} round. The 0^{th} (63^{rd}) Sbox refers to the 5 MSB (LSB) from each word in that round. The i^{th} Sbox from round n will be referred as S_i^n. The bit at round n, row r and column c will be represented as $s_{r,c}^n$. We refer to [DEMS19] for more information on the Ascon authenticated encryption (AE) scheme.

3 Using CP to find differential characteristics

3.1 Existing heuristics to find linear characteristics

In [DEM15], the authors created a heuristic tool to find linear characteristics, mainly with applications to many candidates of the CAESAR competition, including the Ascon permutation. We provide a brief summary of the guess-and-determine search algorithm:

1. Choose a partial characteristic as a starting point. This may come from some other algorithms or the cryptanalyst's intuition.
2. Choose a guessable item X' and fix it to be a valid guess x' based on the search strategy, where X' can be a single bit or an Sbox. Then, based on the guess x', propagate through X' neighboring bits through various rounds. This may fix some other bits in the process or it may lead to a contradiction which indicates that x' cannot be chosen at this point as it will lead to an impossible transition.
3. If fixing $X' = x'$ does not cause a contradiction, continue to choose another guessable item depending on the search strategy.

4. If fixing $X' = x'$ causes a contradiction, we switch to another valid guess for X'. If no valid guess is available for X', backtrack to the item we guessed before X' and choose another valid guess for it.
5. If all the bits are fixed in the end, a linear characteristic is obtained, else the characteristic at the starting point is invalid.

Depending on the search strategy, the algorithm can be restarted after several runs. While this tool is used to find linear characteristics, this guess-and-determine framework can be used for finding different characteristics as well. We find this algorithm to be very similar to the inner workings of the heuristics that most CP solvers employ. Thus, we decided to use CP to simulate this type of dedicated heuristic tool.

3.2 A brief introduction to CP

Constraint programming (CP) is a declarative framework, in which the programmer describes the problem and leaves the resolution to a solver. The problem is described in terms of *variables* on given *domains*, which are given values by the solver during the resolution. These variables are linked together by *constraints*, which correspond to rules the solver must follow. Finally, an *objective function* to be maximized or minimized can be defined, otherwise, it can also be a satisfaction problem, finding all possible solutions. Cryptanalysis problems, such as the search for differential characteristics, are easily modeled with variables representing the characteristic itself, constraints enforcing the propagation rules and an objective function maximizing the overall probability. Other declarative frameworks, such as MILP [SHW$^+$14] and SMT [AK18] are frequently used for similar problems. While there is currently no definitive consensus on which framework is the fastest [SGL$^+$17], for the purpose of this work we will focus on CP. One particular advantage of CP that we rely heavily on is its support for *table constraints*: Let x, y, z be 3 variables, and T be a list of allowed values for x, y, z, then $\text{Table}([x, y, z], T)$ enforces the constraint that the triplet, x, y, z can only take up one of the values listed in T. This is particularly useful to represent non-linear transitions. Furthermore, CP modeling languages such as MiniZinc allow for a fine-grained definition of search heuristics, which have a great impact on the resolution speed. These heuristics are composed of variable selection heuristics, defining the order in which the variables are considered by the solver, and value selection heuristics, defining which value to try first for the variables. For cryptanalysis problems, a logical choice for value selection is to start with 0, since characteristics with high probability tend to have low numbers of active positions.

The first application of CP to the search of differential characteristics was [GMS16], where the authors use CP to find optimal related-key differential characteristics on the AES cipher faster than previous works. The authors exhibited a 4-round related-key differential characteristic that was better than the one previously thought to be optimal. The search is decomposed into two parts, following the method applied in previous work: in a first step, the word variables are abstracted to bits (1 for a non-zero word, 0 otherwise), and a solver is used to find truncated characteristics with a minimal number of active Sboxes. These characteristics are then checked in a second step, where a solver attempts to assign non-zero byte values to the positions of the active words found in step 1. Solutions that pass step 1, but not step 2, are referred to as *inconsistent*. The model contains a more advanced constraints on linear incompatibilities in step 1, effectively filtering out most of the inconsistent solutions. In [GLMS18], the authors showed that the resolution speed for this problem could sometimes be improved by decomposing step 1 into independent sub-problems, defined by the repartition of the active Sboxes per round. Finally, in [GLMS20], the same authors proposed an extended model, with more advanced filtering of inconsistent solutions in step 1 by exploiting the linear parts of the key schedule.

3.3 CP modeling

Most of the research on CP models for differential characteristics focus on word-oriented block ciphers, rather than ciphers with a bit-oriented linear layer such as Ascon. While the general modeling techniques we used are rather straightforward and derived from techniques used for word-based ciphers, tackling the large state size of Ascon is challenging.

As mentioned in Section 3.1, we find the strategies used in the guess-and-determine algorithm to be very similar to that of CP. Thus, in order to automate the process, we decided to model the search for differential characteristics in Ascon permutation into a CP problem. Then we use CP solvers to find good differential characteristics. The advantages of doing so are clear: a cipher can be very easily modeled (saving cryptanalyst's time and reducing the risk of having incorrect results), we can tap into the vast number of strategies given by various CP solvers. The language we chose for modelling was `MiniZinc` [NSB+07]. We have run our program with various solvers, including `Chuffed` [CSS+], `OR-tools` [PF], `choco` [PFL16] and `Gecode` [STL19]. Since our problem has a large search space, the solvers cannot finish searching the entire space in practical time. Thus, the efficiency of the solver is crucial. The efficiency of a solver in solving a problem varies depending on the model and problem. Eventually, we decided to go with `Chuffed` as it returns the best results faster than the other solvers for some of the test cases we had. We note that in this section, we did not obtain better differential characteristics than what the Ascon designers found, but we propose a new and easier method for finding a differential characteristic for the Ascon permutation. In the following paragraphs, we will explain our model of Ascon in CP. To encourage the research into CP methods, we have provided our codes (including the model) at `https://git.io/J0AM9`.

During the development of our models, we experimented with several modeling choices that did not result in better performances. Some of our previous choices include using a combination of the Ascon and its inverse permutation as well as representing the DDT using table constraints with all integers. However, these models have some issues. Firstly, p_L^{-1} has much higher diffusion capabilities than p_L, causing each constraint to involve more variables and therefore more branches in the search tree. Next, since `MiniZinc` does not support bitwise operations such as rotations and XOR on integer variables, we have to convert between the integers and their binary representation when we have to deal with p_L. This slows down the search process. Thus, the choice of using a binary representation for all operations is preferred, despite the large state required as we do not have to convert between integer and its binary representation. The only exception is the probabilities for the Sbox representation. Since we do not need to use them for any binary operations, we retain them in integer form.

Objective function. We consider probability variables $pr_{n,i}$ representing the negative base 2 logarithm of the transition probability through the Sbox for all round n and Sbox position i. Our objective function is therefore

$$\text{Minimize} \sum_{n,i} pr_{n,i}.$$

State. In round n of Ascon permutation, we have two 2D arrays to represent the state *before-substitution* (BS_n) and *after-substitution* (AS_n) respectively. The p_S layer occurs from BS_n to AS_n and the p_L layer is performed from AS_n to BS_{n+1}.

p_S computation. We represent the DDT of the Sbox as a list of valid tuples, with dimension of 317×11, with each row containing a possible Sbox transition $(\Delta_x \to \Delta_y)$ where $\mathbb{P}(\Delta_x \to \Delta_y) > 0$. We represent Δ_x and Δ_y in their binary form, and the probability as a negative $\log_2$: `x[0..4],y[0..4],`$-\log_2(\mathbb{P}(\Delta_x \to \Delta_y)$`)`. We then enforce the Sbox transitions using a table constraint, for all rounds n and Sbox position i (where $BS_{n,i}$ and $AS_{n,i}$ are 5-bit arrays):

$$\texttt{Table}([BS_{n,i}||AS_{n,i}||pr_{n,i}], \texttt{DDT})$$

For instance, the differential transition, $3 \rightarrow 1$ has a probability of 2^{-3} and thus will be represented as a row in the DDT as `[0,0,0,1,1,0,0,0,0,1,3]`.

p_L computation. In the linear layer, we simply introduce a function, `rRot` and a predicate `xor4`. Note that in this description, a word refers to a single row in the ASCON state, *i.e.* a 1D array with 64 bits. `rRot` takes in a word `w` and an integer `rot_value` and returns `w` rotated by `rot_value` to the right. `xor4` takes in 4 words `w0,w1,w2,op` and checks if the sum of `w0[i],w1[i],w2[i],op[i]` is 0, 2 or 4 for all $i \in \{0..63\}$. This ensures that the output word $\texttt{op} = \texttt{w0} \oplus \texttt{w1} \oplus \texttt{w2}$. Next, we can simply apply the function and predicate as per the operations described for p_L in Section 2.2. As a concrete example, the permutation for the first row is represented as such:

```
constraint forall (n in 0..N-1) ( let {
array[0..63] of var 0..1:w0 = rRot(array1d(0..63,
                                  [AS[n,0,c]|c in 0..63]),19),
array[0..63] of var 0..1:w1 = rRot(array1d(0..63,
                                  [AS[n,0,c]|c in 0..63]),28),
array[0..63] of var 0..1:w2 = array1d(0..63,[AS[n,0,c]|c in 0..63]),
array[0..63] of var 0..1:op = array1d(0..63,[BS[n+1,0,c]|c in 0..63]) }
in xor4(w0, w1, w2, op) );
```

3.4 Search strategy and additional constraints

We would like to highlight that the search strategy is extremely crucial in reducing the number of steps taken per branch. By selecting a good condition as a branch, we can eliminate paths that may not lead to good characteristics quicker, thereby resulting in a more efficient search progression. Most of the solvers support various search strategies and in this paper, we are focusing on the search strategy we used for the solver `Chuffed`.

Initial experiments using naive search strategy[1] were insufficient to obtain good characteristics within a reasonable time (compared to the characteristics found by the designers), so we focused our effort on finding search strategies that permit the solver to find good results. We have also tried to start the search by prioritizing the array $pr_{n,i}$ first followed by the values of $BS_{n,i}$ and $AS_{n,i}$. However, the improvements obtained are still not as good as desired.

The best search strategy we have come up with followed closely with the intuition of how the best characteristic should look like: in general, the best differential characteristics are constructed with just a few active Sboxes in the middle rounds and spread to more active Sboxes at the front and back of the characteristic, we would like our solver to search in that similar fashion. Thus, we would limit the number of active Sboxes in the middle rounds and start our search from the middle round. However, one consideration to take note of would be that the inverse linear layer of the ASCON permutation has more diffusion as compared to that of the forward direction, causing the number of active Sboxes to increase significantly more in the backward direction compared to the forward direction when comparing the same extension in the total number of rounds. As such, for 4, 5 and 6 rounds of the ASCON permutation, we configure our solver to start the search at round 2. Then, we also limit the number of active Sboxes at round 1. This ensures the middle round has a small number of active Sboxes and the backward direction does not explode in

[1] Chuffed has a naive based strategy that relies on activity-based search

terms of number of active Sboxes in round 1. Next, after searching at round 2 and fixing the Sboxes, our configuration will force the solver to search for active Sboxes at round $1, 0, 3$ in that order. We do this by adding additional constraints which retrieve the number of active Sboxes at each round. Once a characteristic has been found, the probability is evaluated for the maximum. To ensure that we have randomization and so as to favor high probability characteristics, the variable selection is set to `random_order` and value selection is set to `indomain_min` (this ensures that we try out the inactive Sboxes first). In addition to that, we also restart the search after every 10000 nodes. Concretely, our search strategy for 4 rounds is as follows:

```
search_ann = seq_search([
int_search(row(sboxes,2), random_order, indomain_min, complete),
int_search(row(sboxes,1), random_order, indomain_min, complete),
int_search(row(sboxes,0), random_order, indomain_min, complete),
int_search(row(sboxes,3), random_order, indomain_min, complete),
int_search(array1d(pr_{n,i}), occurrence, indomain_min, complete)]);
```

where `sboxes` is an array showing the positions of active sboxes and `occurrence` helps to find the variable with the smallest domain. To come up with a generic framework and reduce the search space, we use a method similar to that of [GLMS18]. We can generalize the method for finding differential characteristics using CP as a two-step process:

1. Using just a single linear layer, we find all possible active Sbox transitions (up to symmetry) that lead to k active Sboxes after one round. We fix the 63^{rd} Sbox to be active to eliminate most of the symmetry by rotation.
2. Using the full N round permutation model, force round n and $n + 1$ to have the transition in Step 1.

Of course, the above method could only work efficiently if Step 1 does not lead to an explosion in the number of characteristics. Thus, k has to be experimentally tested to ensure feasibility depending on the cipher.

3.5 CP results

To find differential characteristics for the Ascon permutation, in Step 1 of the method described in Section 3.4, we set $k = 2, 3$ and 4 and let it run for all possible transitions; in Step 2 we tried out for $4, 5, 6$ rounds. For Step 1, we can exhaust the total number of possible transitions: we have the total number of possible transitions at $9, 155, 1776$ for $k = 2, 3, 4$ respectively. For Step 2, we run the program with a time limit of 1.5 hours for each possible transition respectively. Note that every possible transition is independent and thus, they can be run in parallel. The best characteristics obtained can be found in Table 2. Note that while we did not find better differential characteristics than what was reported by the designers for rounds 4 and 5, this shows that by using CP, we can achieve similar results to dedicated heuristic algorithms.

Throughout the rest of this paper, we use variations of this base model to search differential characteristics with specific properties, by including additional constraints. These constraints include, for instance, restrictions on the first row for the forgery scenario. When such modifications are included, they will be described in their corresponding section.

4 Ascon permutation distinguishers

To distinguish a specific permutation $\mathcal{P}$ over a random permutation $\mathcal{R}$, we aim to obtain an algorithm that can detect a certain property with a higher probability for $\mathcal{P}$ (or its

Table 2: The best probabilities obtained using CP with various number of rounds N and the number k of active Sboxes at round 2. The probabilities are given in their $-\log_2$ value.

N \ k	2	3	4
4	141	107	127
5	209	190	198
6	> 320	305	> 320

inverse) that an adversary could achieve with a black-box access to $\mathcal{R}$ (which we call a "distinguisher" $\mathcal{D}$ for $\mathcal{P}$). The difference in probability is also known as the advantage i.e.

$$Adv(\mathcal{D}) = |Pr[\mathcal{D}(\mathcal{P})] - Pr[\mathcal{D}(\mathcal{R})]|$$

Theoretically, a distinguisher will work as long as $Adv(\mathcal{D}) > 0$. However, for practical reasons (some constant factors not being exactly taken into account), we would like to have the advantage to be higher than a certain threshold value.

4.1 Distinguishers for unkeyed permutation and keyed permutation

We first propose a categorization for distinguishers. The motivation for doing so comes from the difference between an unkeyed and keyed permutation. In the case of the former, such as for hash functions or under the known-key and chosen-key model, we can treat the cryptographic primitive as a non-black-box: one is allowed to conceive distinguishers that start-in-the-middle, and then propagate forward and backward in an inside-out fashion, utilizing the degrees of freedom to reduce as much as possible the computational complexity. In the case of a keyed permutation, the attacker has to treat the primitive as a black-box: starting the analysis either from the first or the last round. In most symmetric-key ciphers, the key is applied before the nonlinear operation, thus making it difficult to utilize any degree of freedom to reduce the cost. Consequently, comparing these two types of distinguishers under the umbrella term "distinguishers" is not exactly fair as they are often useful in different contexts and most of the non-black-box have a much lower cost compared to black-box distinguishers.

4.2 Obtaining constraints from a differential characteristic

Constraints from p_S. To utilize the degrees of freedom, we have to first locate what are the constraints/equations. Since the only non-linear operation is the substitution layer, all of the constraints lie within the active Sboxes in the various rounds. These equations can be obtained by observing how the differences interact with the AND gates in these Sboxes. For illustration purposes, we propose first an example. Let Δ_{in} and Δ_{out} be the input and output differences of a Sbox respectively. Since the AND gates are the only non-linear ones, we can propagate the Δ_{in} and Δ_{out} until we obtain the required difference at the input and output of the AND gates. We will use the input difference $\Delta_{in} = \texttt{0x6}$ and output difference $\Delta_{out} = \texttt{0x1}$ as an example and it is shown in Figure 2. To describe in detail, we label the AND gates in a vertical order in Figure 2, the top AND gate being AND_0 and the bottom-most AND gate being AND_4. For each AND gate, we call the input with the NOT gate as inp1 and the other to be inp2. Since $\mathbb{P}(\Delta_{in} = \texttt{0x6} \rightarrow \Delta_{out} = \texttt{0x1}) = 2^{-4}$, we will expect 4 independent equations. Based on the inputs and output of an AND gate, linear constraints with respect to the input bits can be obtained. For instance, AND_1 has an active output bit difference which requires the inactive input bit, inp1_1, to be equal to 1. Thus, we can deduce a constraint

$x_1 \oplus 1 = 1$. For AND_0, there are no active input bits which means there is no constraint here. For AND_3, we will need the inputs inp1_3 and inp2_3 to have alternate signs. Thus, $\text{inp1}_3 \oplus \text{inp2}_3 = 1 \implies (x_3 \oplus 1) \oplus (x_3 \oplus x_4) = 1 \implies x_4 = 0$. Working out the remaining 2 AND gates, we will get the remaining 2 independent equations.

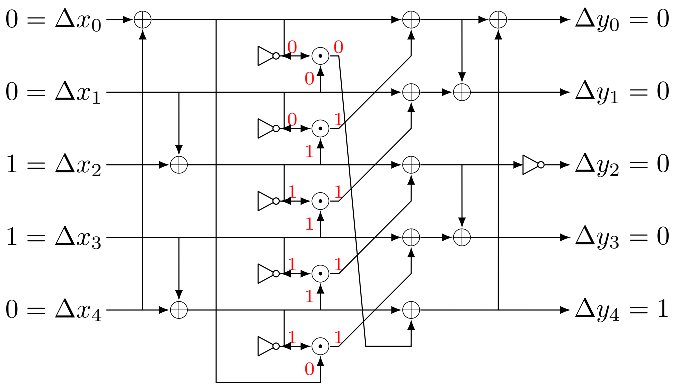

Figure 2: ASCON Sbox with the propagation of $\Delta_{in} =$ `0x6` and $\Delta_{out} =$ `0x1`

Propagating constraints through p_S and p_L. Unlike a black-box distinguisher, a non-black-box distinguisher may start anywhere in the middle. Thus, we have to pick a particular round (including those in the middle) to spend our degrees of freedom. As the constraints are in different rounds (depending on where are the active Sboxes), we need to propagate the constraints to that particular round before spending our degrees of freedom. Note that we also must ensure that the set of constraints we chose to spend represents an independent set too. As constraints may not necessarily be linear after propagating through an Sbox, we have to simplify the computations. For these non-linear constraints, we propose two methods to resolve them. For constraints involving the same Sbox, we can spend all 5 bits to fix the entire Sbox; this basically fixes the input/output value such that the expected difference transition is indeed enforced. Another method is that when we propagate constraints through the Sboxes, we only keep the linear constraints. For the linear layer, we can simply apply the changes according to p_L.

4.3 Limited-birthday distinguishers for the ASCON permutation

Limited-birthday distinguishers were first introduced in [GP10] to obtain a distinguisher under the known-key setting for 8-round-reduced AES-128 as well as 8-round-reduced Grøstl-256. Formally, this distinguisher is built upon the limited-birthday problem (the attacker is limited in his ability to apply the birthday search): given a l-bit permutation F, and $D_{in}, D_{out} \subseteq \mathbb{F}_2^l$, we would like to generate a pair of inputs (x, x') where $x \oplus x' \in D_{in}$ will lead to an output pair $(F(x), F(x'))$ such that $F(x) \oplus F(x') \in D_{out}$. For an ideal permutation F, the best complexity for obtaining a right pair is given by [IPS13]:

$$C(|D_{in}|, |D_{out}|) = \max\left\{\min\left\{\sqrt{\frac{2^{l+1}}{|D_{in}|}}, \sqrt{\frac{2^{l+1}}{|D_{out}|}}\right\}, \frac{2^{l+1}}{|D_{in}||D_{out}|}\right\} \tag{1}$$

Black-box limited-birthday distinguisher. To obtain the black-box limited-birthday distinguisher, we simply take a differential characteristic and allow the single unique output difference to spread for one more round with probability 1 (i.e. we allow all the possible differential transitions for all the active Sboxes in the last round). By doing so, we have $|D_{in}| = 1$ and $|D_{out}|$ to depend on the specific characteristic. On the other hand, we have to ensure that the resulting cost of the distinguisher is lower than that of the generic complexity, using Equation 1. We recall that for black-box distinguisher, will not use degrees of freedom to reduce the computational complexity. Thus, the best distinguisher will be the one built upon the characteristic with the highest probability. Using CP, with the search for active Sboxes from round 0, we have obtained a differential characteristic over 3 rounds with a probability of 2^{-40} (Table 13). After allowing the output difference to spread freely for one more round, we obtained a 4-round limited-birthday distinguisher with $|D_{out}| = 2^{115}$. The generic computational cost for generating a pair with a similar structure is $2^{102.5}$ primitive calls. For the 4-round differential characteristic, we are unable to use the best known characteristic of 2^{107} as the corresponding generic complexity is lower than that (when allowing the output difference to develop freely after one round). Thus, we decided to search for one within the set of results we have obtained from Section 3 and we managed to find another characteristic (LB4.1) with a probability 2^{-109}, while the corresponding generic complexity is 2^{230} primitive calls. Note that since we are dealing with a permutation, we have also searched with the inverse function (i.e. keeping $|D_{out}| = 1$ and spread backward). LB4.1 can be found in Appendix B. Unfortunately, we are unable to get black-box limited-birthday distinguishers for a higher number of rounds due to the low probabilities of the differential characteristics.

Non-black-box limited-birthday distinguisher. When one is allowed to start from the middle, the attack complexity will significantly be reduced, being lower than the generic one. In Figure 3, we provide a pictorial view of how we build our non-black-box limited-birthday distinguishers. We first consider a differential characteristic on N rounds and then we allow the input and output differences of the differential characteristic to spread backward (over $b = 1$ rounds) and forward (over $f = 1$ rounds) with probability 1 respectively (similar to what was conducted in [DGPW12]). In this case, D_{in} and D_{out} will represent the sets of possible differences at the start and the end after the spreading of the differences. A suitable differential characteristic has therefore two requirements. Firstly, it should not have a high number of active Sboxes at the start, or else the backward diffusion may result in a large set of D_{in}. Secondly, we would like our active Sboxes to cluster at a single round (or neighboring rounds) so that we are able to use fewer bits on the constraints. Suppose we choose to spend the degrees of freedom at the n^{th} round, the constraints at round $n' \neq n$ will require more than one bit to control. The number of bits required increases as $|n - n'|$ increases. While ASCON's designers have provided good differential characteristics in [DEMS19], they might not the best for constructing limited-birthday distinguishers. The designers of ASCON gave a 4-round and 5-round differential paths that have a high number of active Sboxes at the first and last round, which increases the average number of bits required to fix a constraint. Thus, to maximize the number of constraints to control, during the search for characteristics using CP, we choose to remove the idea of starting from the middle, but instead to just start the search at round 0 instead. This allows the search to fix the smallest number of active Sboxes at round 0, which usually implies that there are more active Sboxes at the end of the characteristic. The characteristic using this search method usually satisfies the criteria we want. We obtained 2, 3, 4 and 5-round differential characteristics that can be found in Appendix B. To differentiate these characteristics from the best ones, we will call them LBn where n is the number of rounds. For each of these differential characteristics, we extend forward and backward by one round with probability 1 (allowing the differences to spread freely), to form limited-birthday distinguishers for 4, 5, 6 and 7 rounds.

Choosing the bit/Sbox to fix the constraint. As mentioned previously, we have to propagate some of the constraints to a particular chosen round, n^*. As we propagate constraints through p_S and p_L, the number of bits at n^* influencing the constraints increases.

After obtaining the differential characteristic, we exhaust all possible positions and found out that choosing to spend the degrees of freedom at round $n - \frac{3}{2}$ for a n-round differential characteristic results in the best distinguisher. That is, if $P = p_L \circ p_S \circ p_{AC} \circ p_L \circ P'$, the position is at the end of P'. Since we need to ensure the constraints are independent, also we need a strategy to choose a sequence of what constraints to be fixed first. In our case, since $n^* = N - \frac{3}{2}$, we follow a general way to compute. First, we fix the entire Sbox for those constraints at round $n^* - \frac{1}{2}$. Next, we list the remaining constraints in a list, prioritized based on the round number (we favor $n^* + \frac{1}{2}$ over round $< n^* - \frac{1}{2}$) followed by the number of linear bits or Sboxes that we are still free to change. In Appendix A, we show an example to illustrate this strategy.

Computing the advantage over an ideal permutation. After spreading the differential characteristic forward and backward, we can evaluate the size of D_{in} and D_{out} and thus, compute the generic complexity using Equation 1. We can compare it to that of the limited-birthday distinguisher: $2^d/\mathbb{P}(\text{differential characteristic})$, where d is the number of independent constraints that we have fixed. Details of the distinguishers' parameters and generic complexities are summarized in Table 3.

Table 3: Parameters of the limited-birthday distinguishers for Ascon permutation (320 bits).

	Trail	# Rounds	$\lvert D_{in}\rvert$ ($\log_2$)	$\lvert D_{out}\rvert$ ($\log_2$)	Comp. ($\log_2$)	Generic comp. ($\log_2$)
non black-box	LB2	4	169.98	32	1	119.02
	LB3	5	169.98	115	2	75.51
	LB4	6	169.98	175	9	73
	LB5	7	213.42	180	53	70.5
black-box	LB3	4	0	115	40	206
	LB4.1	5	0	91	109	230

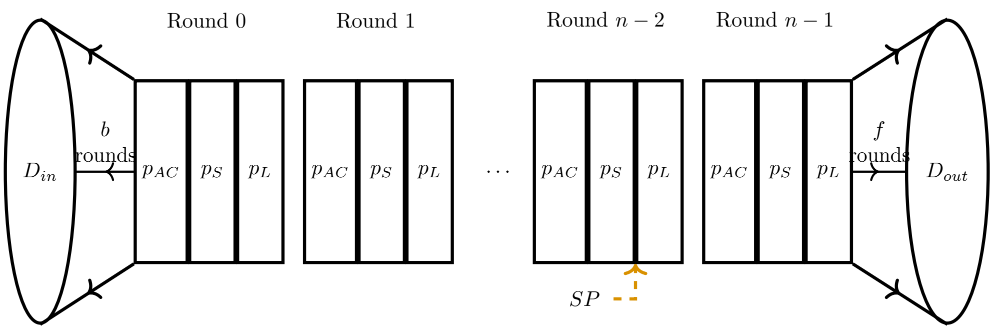

Figure 3: An illustration of how we construct our non-black-box limited-birthday: after obtaining a differential characteristic, we allow it to spread backward and forward by b and f rounds respectively with probability 1. SP indicates the starting point of our limited-birthday distinguisher.

Improved limited-birthday distinguishers. As the gap of the limited-birthday distinguishers and their respective generic complexity is large, we can explore the possibility of spreading the differences forward by 2 rounds (i.e.$f = 2$). However, the computational cost required to accurately calculate the all the possible differences is high. Thus, we decided to find a lower bound by estimating the number of impossible differences (an upper bound on $|D_{out}|$). In order to do so, we keep track of the bits that are inactive. For example, with the input difference of `0x04`, the possible Sbox output differences are `0x06`, `0x0e`, `0x16` and `0x1e`. Thus, the active bit positions can be located by the OR operation, $OR(\texttt{0x06}, \texttt{0x0e}, \texttt{0x16}, \texttt{0x1e}) = \texttt{0x1e}$. In other words, the LSB is definitely inactive. In the next p_S, we can compute the impossible differences. For example, given the active bit positions `0x1c`, the possible input differences are `0x04`, `0x08`, `0x0c`, `0x10`, `0x14`, `0x18`, `0x1c`. According to the DDT, the output differences `0x00`, `0x02`, `0x11` and `013` are impossible for any of those input differences. Changing the operation of XOR to OR in p_L can simulate the spread of active bits for the linear layer. Using this technique, we can improve most of the limited-birthday distinguishers. Some of the characteristics we previously used are no longer the best characteristic for this particular technique and thus, we did a search for one within the set of results we have from Section 3. The summary of the results can be found in Table 4 and the characteristics can be found in Appendix B.

Table 4: Parameters of the improved limited-birthday distinguishers on the ASCON permutation (320 bits).

	Trail	**# Rounds**	$\lvert D_{in}\rvert$ ($\log_2$)	$\lvert D_{out}\rvert$ ($\log_2$)	**Comp.** ($\log_2$)	**Generic comp.** ($\log_2$)
non black-box	LB2	5	169.98	116.19	1	75.51
	LB3	6	169.98	$\leq$ 290.64	2	$\geq$ 15.18
	LB4.2	7	213.42	$\leq$ 246.72	34	$\geq$ 37.14
black-box	LB2	4	0	$\leq$ 116.19	8	$\geq$ 204.81
	LB3.1	5	0	$\leq$ 160.77	65	$\geq$ 160.23

4.4 Rectangle distinguishers for the ASCON permutation

Boomerang distinguishers [Wag99] can be seen as a variant of the differential distinguisher. The idea is to use two short differential characteristics instead of a single long one to cover the rounds of a function. Let E be a l-bit permutation and suppose it can be broken down into two independent parts $E = E_1 \circ E_0$. Now, suppose again that there exists a differential characteristic for E_0, T_{E_0} (upper characteristic) with difference propagation $\alpha \to \beta$ with a probability of p, and a characteristic for E_1, T_{E_1} (lower characteristic) with difference propagation $\gamma \to \delta$ with a probability of q. Assuming that $pq \gg 2^{-l/2}$, then the following algorithm can distinguish E from a random permutation:

1. Generate $(pq)^{-2}$ unique plaintext pairs (P_0, P_1) such that $P_1 = P_0 \oplus \alpha$.
2. For each plaintext pair (P_0, P_1), compute $C_0 = E(P_0)$ and $C_1 = E(P_1)$
3. Compute $C_2 = C_0 \oplus \delta$ and $C_3 = C_1 \oplus \delta$
4. Ask for the decryption of C_2 and C_3, i.e. $P_2 = E^{-1}(C_2)$ and $P_3 = E^{-1}(C_3)$
5. If $P_2 \oplus P_3 = \alpha$, return "E"
6. If all pq^{-2} pairs do not satisfy $P_2 \oplus P_3 = \alpha$, return "random"

Assuming that the characteristics T_{E_0} and T_{E_1} are independent, the boomerang attack will succeed with a probability of $(pq)^2$. Rectangle attack [BDK01] is an improvement over the original boomerang attack. Instead of using just a single characteristic for E_0, one can use multiple characteristics that all start with a difference of α but that end with different output differences. Similarly, for E_1, we use characteristics that have different input differences but with a single output difference δ. The probability evaluation is usually more complicated as there are many characteristics involved. For characteristics that are not computationally verifiable, probability estimation can still be done experimentally by only considering the last few rounds of the upper characteristic and the first few rounds of the lower characteristic.

Construction of black-box rectangle distinguishers for the ASCON permutation. To create rectangle distinguishers for the ASCON permutation, we use LB3. For the upper characteristic, we use the first 2 rounds of $\mathrm{LB3}_{\lll 31}$ (left rotation of LB3 by 31 bit positions) and for the lower characteristic, we just use LB3 as it is. Together, they form a 5-round rectangle distinguisher for the ASCON permutation. For completeness, we give the entire characteristic in Table 25 in Appendix E. To verify the compatibility of the characteristics as well as to get a better probability estimation, we experimentally verified a shortened sub-characteristic: from the start of the upper characteristic to the second round of the lower characteristic. After searching for 1000 right quartets, the average number of quartets tested was $12177 \approx 2^{13.57}$. Thus, we estimate that the 5-round boomerang distinguisher has a complexity of $4 \cdot 2^{13.57} \cdot 2^{32 \cdot 2} = 2^{79.57}$ primitive calls. Note that this also means that we have a rectangle distinguisher on 4 rounds with $4 \cdot 2^{13.57} = 2^{15.57}$ primitive calls. The reason for the rotation value is simple: since a differential characteristic for the ASCON permutation can be rotated without affecting the probability of the characteristic, we have up to 64 different possible combinations to form our upper and lower characteristic. We exhausted all the possibilities experimentally and took the best among them with regards to their respective probability.

Construction of non-black-box rectangle distinguishers. For non-black-box distinguishers, we use the same upper and lower characteristic as the one in the black-box. For the 4-round non-black-box rectangle distinguisher, we choose to start at round 3. This allows us to control all the 3×2 constraints in round 3. We choose not to control the linear constraint from round 2 as it has the ladder switch effect [BK09] (i.e. probability 1). This reduces the cost of the distinguisher from $2^{15.57}$ to $2^{9.57}$ primitive calls. For the 5-round non-black-box rectangle distinguisher, we choose to start at round 4. This allows us to control 32 constraints in round 4 and 3 linear constraints (1 constraint per Sbox) in round 3. This reduces the cost from $2^{79.57}$ to $2^{44.57}$ primitive calls. We show a pictorial view of the starting point for the distinguisher in Figure 4.

4.5 Remarks on zero-sum distinguishers

The zero-sum property was first proposed by Aumasson and Meier in [AM] to construct a distinguisher for the permutation used in the KECCAK hash function. For a given permutation $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$, the idea is to create a set of inputs, Z, such that

$$\bigoplus_{z_i \in Z} z_i = \bigoplus_{z_i \in Z} F(z_i) = 0$$

There exists a stronger notion of zero-sum distinguisher called the zero-sum partitions [BC10b]: for a permutation function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we want to find 2^{n-k} disjoint sets, $Z_0, Z_1, ..., Z_{n-k} \in \mathbb{F}_2^n$ such that

- $\bigcup_{j \in \{0 \ldots 2^{n-k}\}} Z_j = \mathbb{F}_2^n$
- $\bigoplus_{z_i \in Z_j} z_i = \bigoplus_{z_i \in Z_j} F(z_i) = 0, \forall\ j \in \{0..2^{n-k}\}$

Zero-sum distinguishers will have a complexity directly linked to the algebraic degree of F. In [BCC11], a very expensive zero-sum distinguisher on full round KECCAK was given. Since the algebraic degrees of the Sboxes in KECCAK and ASCON are the same (forward: 2, backward: 3), one can expect a similar cost for a zero-sum distinguisher against ASCON permutation.

While there is a zero-sum distinguisher proposed on 12-round ASCON permutation in [DEMS15], we can use the same calculations to get an estimated distinguisher for the

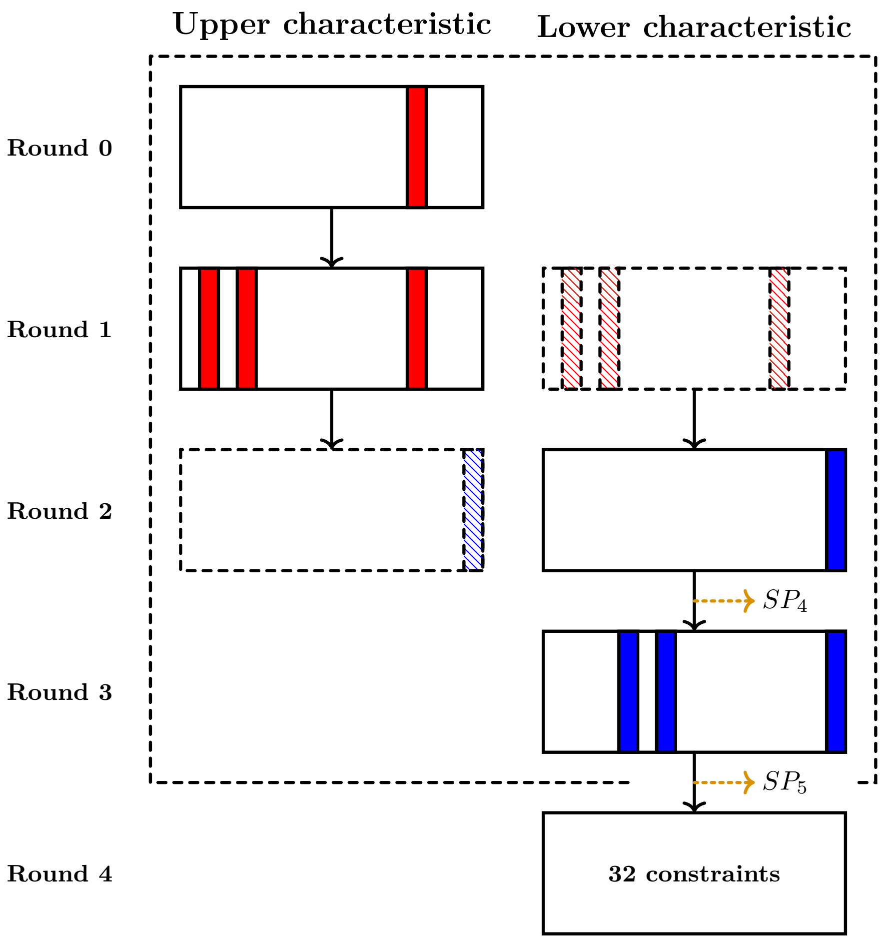

Figure 4: An illustration of the starting points for non-black-box rectangle distinguisher for 4 and 5 rounds. The red and blue bars show the rough position of the active Sboxes for the upper characteristic and lower characteristic respectively. The starting point for the 4-round distinguisher, (SP_4), fixes 6 (linear) constraints for the 3 Sboxes in round 3 while the starting point for the 5-round distinguisher, (SP_5), fixes 3 (linear) constraints for the 3 Sboxes in round 3 and 32 (linear) constraints in round 4.

other numbers of reduced rounds. In fact, using Algorithm 2 in [Tod15], we can see that a zero-sum partition can be obtained as well by starting in the middle.[2]

The estimated costs are found in Table 5 for 4, 5, 6 and 7 rounds, which allows us to have a fair comparison with our distinguishers. The best distinguisher for each number of rounds is underlined. We have experimentally verified that they do work for their respective complexities as well.

Single zero-sum distinguishers. While these distinguishers have low complexities, the advantage over the generic attack is extremely small. The generic complexity is measured using the XHASH attack [BM97], and further discussed in [WGR18]. We can try to estimate the cost of finding such a zero-sum for a random permutation. From [WGR18], the cost can be approximated by $M + 2m + 10$, where M is the size of the set of plaintext

[2] Note that a set with the division property of $\mathcal{D}_2^n$ is equivalent to having a single zero-sum. Thus, by starting in the middle, the resulting set has the zero-sum property. The construction can be easily adapted to form zero-sum partitions.

structure and m is the state size. In other words, the advantage of a zero-sum distinguisher on a real permutation as compared to a random permutation is $2m + 10$. Thus, as the size of the set increases, the relative advantage of the zero-sum distinguishers diminishes. When the size of the set is 2^k with $k \geq 10$ for Ascon, the advantage falls under a factor of 2.

Zero-sum partition distinguishers. The method used to construct the zero-sum set can actually construct a zero-sum partition easily (see Proposition 2 of [BC10b]). They have also provided a generic algorithm for finding a zero-sum partition of size 2^k. To our knowledge, this is the best known algorithm for finding a zero-sum partition and the complexity is $\mathcal{O}(2^n)$. However, this does not mean that the algorithm is the best possible one. This is in contrary to limited-birthday distinguishers where a lower bound is proven in [IPS13]. Furthermore, the verification for a full zero-sum partition requires the computation of $2^{n-k} \times 2^k = 2^n$ inputs as well. This might be overcome with a sampling of the final partition by the verifier, but it renders the advantage of zero-sum partitions compared to potential generic attacks quite unclear at the time of writing. Thus, to have a fair comparison of the generic complexities of zero-sum distinguishers, we are comparing against the single zero-sum distinguishers in Table 6.

Table 5: Number of bits required to vary in the zero-sum distinguishers using the same computations as in [DEM15]. "One free round" refers to the technique from Boura *et al.* [BC10a] and used in [DEM15] to add one round in the middle of the basic distinguisher. "Division property" was calculated based on the Algorithm 2 from [Tod15].

No. of rounds	**3**	**4**	**5**	**6**	**7**
Basic distinguisher	5	9	<u>10</u>	17	28
One free round		<u>5</u>	<u>10</u>	<u>10</u>	20
Division property		<u>5</u>	<u>10</u>	<u>10</u>	<u>18</u>

4.6 Results on distinguishers

In Table 6, we have listed our limited-birthday and rectangle distinguishers with their complexities alongside other known distinguishers for the Ascon permutation. We remark that compared to other non-black-box distinguishers on the same number of rounds of the Ascon permutation, our limited-birthday distinguishers have the smallest complexity for 4, 5 and 6 rounds. The parameters and generic complexities are given in Table 3. Some conforming pairs for the limited-birthday distinguishers (except LB3.1) and 4 rounds of rectangle distinguisher can be found in Appendix C.

5 Forgery attacks on Ascon-128 and Ascon-128a

As Ascon is an AE family, authenticity is one of the key requirements for the schemes to be secure. In this section, we present some forgery attacks against the (reduced-round) iteration as well as finalization phases of the AE schemes. We rely again on our new CP model to search for differential paths with additional constraints for this particular scenario.

5.1 Additional constraints to the CP model

During the iteration phase, one can only access the rate part of the duplex construction to get a forgery. For Ascon-128 and Ascon-128a, the rate part refers to the first row of

Table 6: Summary of distinguishers against the ASCON permutation. The reported complexities in this table are expressed in terms of the number of primitive calls, which explains some of the small variations one can observe in some numbers when compared to the original values.

Black-box distinguishers				
# rounds	**Type**	**Complexity** ($\log_2$)	**Generic comp.** ($\log_2$)	**Ref.**
4	Diff.-Linear	2	—	[DEMS15]
	Rectangle	15.57	320	This paper
	Integral	5	—	[RHSS21]
	Differential	108	320	[DEMS21b]
	Linear	101	—	[DEMS15]
	Limited-birthday	8	≥ 204.81	This paper
5	Rectangle	79.57	320	This paper
	Integral	16	—	[RHSS21]
	Differential	191*	320	[DEMS21b]
	Linear	189	—	[DEMS15]
	Limited-birthday	65	160.23	This paper
	Truncated Diff.	108	—	[Tez16]
6	Integral	31	—	[RHSS21]
7	Integral	60	—	[RHSS21]

Non-black-box distinguishers				
# rounds	**Type**	**Complexity** ($\log_2$)	**Generic comp.** ($\log_2$)	**Ref.**
4	Rectangle	9.57	320	This paper
	Zero-sum	5	9.41**	[DEMS15]
	Limited-birthday	1	119	This paper
5	Rectangle	44.57	320	This paper
	Zero-sum	10	10.7**	[DEMS15]
	Limited-birthday	1	75.51	This paper
6	Zero-sum	10	10.7**	[DEMS15]
	Limited-birthday	2	≥ 15.18	This paper
7	Zero-sum	18	18.00*	[Tod15]
	Limited-birthday	34	≥ 37.14	This paper

* We have confirmed with the authors that their 2^{-193} reported differential path can be optimized to 2^{-190} with a better selection of some Sbox differential transitions.
** The generic complexities computed for the zero-sum distinguishers are for the single zero-sum distinguisher. More explanation can be found in Section 4.5.

the state, and the first and second rows respectively. In our tool, we limit the search to just a single block of p^b and hope to inject a difference in the rate part of the input and output of that block. To do so, we add the following constraint to our CP model (we use ASCON-128 as an example):

```
constraint sum (c in 0..63, r in 1..4) (BS[0,r,c]) = 0;
constraint sum (c in 0..63, r in 1..4) (AS[N,r,c]) = 0;
constraint Xor3(stateend[0,0,0..63],
```

```
            stateend[0,4,0..63],statestart[0,0,0..63]);
constraint forall(i in 0..63) (stateend[0,1,i] = statestart[0,0,i]);
constraint forall(i in 0..63) (stateend[0,2,i] = 0);
```

The first and second constraints ensure that there is no difference in the third to fifth row for the input and output of the permutation block as we do not have access to the capacity. The remaining constraints are only for Ascon-128 and they follow Observation 1 from [ZDW19]. In addition to that, we construct a new DDT for the first (resp. last) round: we can only consider differences that start (resp. end) with 0 or 16 for Ascon-128 (0, 8, 16, 24 for Ascon-128a). As for the search strategy, we decided to not just start the search at a single particular round only but to try all possible permutations. For instance, for $n = 2$, we use 2! strategies: search for active Sboxes in the first round, followed by the second round. The second one starts the search at the second round, followed by the first. For $n = 3$ and 4 we have 3! and 4! strategies. For each instance, we allow it to run for 5 days.

For the iteration phase, we can then simply perform a forgery attack as follows:

1. Generate a message, $m = (m_0, m_1)$
2. Ask the oracle for the encryption of m and obtain the corresponding tag T
3. Apply the difference to m_0 and m_1 to get m'_0 and m'_1 respectively
4. Ask the oracle for the decryption of (m'_0, m'_1) with the tag T
5. If the oracle returns $\perp$, repeat from Step 1 with a different message

For the finalization phase, we change the formulation of the DDT for the last round. Since we are only interested in the last two rows of the state (namely rows 3 and 4) as these are the bits that will contribute to the final tag T. Thus, for each Sbox, we can combine the probabilities for the differential transitions with the same input difference and the same output difference truncated to the last two bits. For instance, we have

$$\mathbb{P}(\texttt{0x4} \to \texttt{0x6}) = 2^{-2} \qquad \mathbb{P}(\texttt{0x4} \to \texttt{0x16}) = 2^{-2}$$
$$\mathbb{P}(\texttt{0x4} \to \texttt{0xe}) = 2^{-2} \qquad \mathbb{P}(\texttt{0x4} \to \texttt{0x1e}) = 2^{-2}$$

These can be combined to a single transition for the last round: $\mathbb{P}(\texttt{0x4} \to \texttt{0b***10}) = 1$. We can do the same for the rest of the transition and a new table constraint for it.

The attack on the finalization is then similar to that of the iteration phase:

1. Generate a message, m
2. Ask the oracle for the encryption of m and obtain the corresponding tag T
3. Apply the planned difference to m and T to get m' and T' respectively
4. Ask the oracle for the decryption of m' with the tag T'
5. If the oracle returns $\perp$, repeat from Step 1 with a different message

The best characteristics obtained using the CP program can be found in Table 7. Some of these characteristics we will be using for the attacks.

5.2 Differential improvement by combining differential characteristics

Our CP model is searching for differential characteristics, not differentials. Thus, to improve the probability of the attack, we can combine differential characteristics with the same input difference and output (truncated) differences to form a differential. To achieve this, we first find the best differential characteristic using CP with the additional

Table 7: Probabilities (in $-\log_2$) of the best characteristics found with additional restrictions on the capacity part of Ascon-128 and Ascon-128a.

Primitive	**Target**	**2 rds**	**3 rds**	**4 rds**
Ascon-128	Iteration	156	231	253
	Finalization	10	32	100
Ascon-128a	Iteration	44	116	199
	Finalization	4	19	100

constraints mentioned above. Next, we take the difference of the first and last round of the best characteristic and add it as a constraint in the model. We also change our minimization problem to a satisfaction one: we will enumerate all the possible solutions to our problem. We also add a lower bound to the probability of the characteristics to prevent an overwhelming number of solutions.

5.3 Results

Ascon-128**.** For forgery attacks against the finalization of round-reduced Ascon-128, the summary of the results can be found in Table 8. The characteristics can be found in Table 23 and Table 24 in Appendix D for 3 and 4 rounds respectively. Note that the complexities are expressed in terms of primitive calls. The forgeries for both rounds are using multiple differential characteristics. For each of these characteristics, we have the same input difference and output (truncated) difference. For instance, in the case of 4 rounds, we have found 4 characteristics with probability 2^{-100}, 4 characteristics with probability 2^{-102}, 16 characteristics with probability 2^{-103}, etc. We limited the search at probability $\geq 2^{-115}$, obtaining a total probability of $2^{-95.61}$. Note that this exceeds the recommended limit of processed data blocks for a single key. For 3 rounds, we limited our search at probability $\geq 2^{-47}$ and obtained a total probability of $2^{-31.76}$. Since in each message we have to do 1 encryption and 1 decryption call, the complexity contains an extra factor of 2.

Ascon-128a**.** For forgery attacks against the permutation in the iteration phase, we have found a differential characteristic for 3 rounds with a probability of 2^{-116}. Note that this also exceeds the limit on the number of processed data blocks for a single key. For finalization, we have a characteristic with a probability of 2^{-19} for 3 rounds. The results are summarized in Table 8. The individual characteristics for the iteration and finalization phase can be found in Table 22 and Table 21 in Appendix D respectively.

6 Improved two-round collision attacks on Ascon-Hash

In [ZDW19], the authors proposed an attack on 2-round Ascon-Hash with a complexity of 2^{125}. We will describe below, this attack, but using our differential characteristic. In their attack, they use a differential characteristic that contains differences only in the first row at start and end. This characteristic has a probability of 2^{-199}. To find this characteristic, they first found a 1-round differential characteristic that ends with difference only located in the first row and satisfying certain other conditions. Then, the target difference algorithm is used to prepend the first round.

Using the best 2-round characteristic we have found (see Table 7), we have 27 active sboxes with a probability of 2^{-54} (54 constraints) in the first round and 28 active sboxes with a probability of 2^{-102} in the second round. This characteristic can be found in Table 26 in Appendix F. Using the same techniques used in [ZDW19], we have a 2-round

Table 8: Forgery attacks against round-reduced Ascon-128 and Ascon-128a.

Primitive	# Rounds	Target	Scenario	Type	Complexity ($\log_2$)	Ref.
Ascon-128	3/12	Finalization	Nonce-respecting	Differential	34	[DEMS15]
		Finalization	Nonce-respecting	Differential	32.76	This paper
	4/12	Finalization	Nonce-respecting	Differential	102*	[DEMS15]
		Finalization	Nonce-respecting	Differential	96.61*	This paper
		Finalization	Nonce-misuse	Cube tester	9	[LZWW17]
	5/12	Finalization	Nonce-misuse	Cube tester	17	[LZWW17]
	6/12	Finalization	Nonce-misuse	Cube tester	33	[LZWW17]
Ascon-128a	3/12	Iteration	Nonce-respecting	Differential	117*	This paper
		Finalization	Nonce-respecting	Differential	20	This paper

* Note that these exceeded the limit on the number of processed data blocks for a single key

attack with a complexity of 2^{103} of hash computations. The results can be found in Table 9 and the attack procedure is as follows:

1. Generate a total of 2^{92} random 2-block messages (M_0, M_1). Apply the hash function and retain all the state values.
2. With a probability of 2^{-54}, a state will satisfy the 54 constraints in the first round of the characteristic. This means that we expect 2^{38} values (M_0, M_1) to satisfy it.
3. Append another 2×2^{64} 1-block messages, M_2 and $M'_2 = M_2 \oplus \Delta_{in}$ to each of the 2^{38} messages and hash them. This results in a total of $2^{64+38} = 2^{102}$ pairs of messages.
4. With a probability of 2^{-102}, a pair of messages will satisfy the constraints for the second round. Thus, we will have, on average, one message pair that satisfies the output difference Δ_{out}.
5. Apply a random message block M_3 and $M'_3 = M_3 \oplus \Delta_{out}$ to the message blocks selected at the end of Step 4 and one directly obtains a collision.

Complexity. The complexity of the attack procedure above is $(2 \times 2^{92}) + (2 \times 2^{102}) \approx 2^{103}$ hash function calls.

Table 9: Summary of collision attacks against Ascon-Hash (Ascon-Hasha).

Primitive	# rounds	Complexity ($\log_2$)	Ref.
Ascon-Hash (Ascon-Hasha)	2/12 (2/8)	103	This paper
Ascon-Hash (Ascon-Hasha)	2/12 (2/8)	125	[ZDW19]

7 Conclusion

In this paper, we have explored the use of CP to model the Ascon permutation and used it to find good differential characteristics for various attack scenarios. First, we show that using CP as a generic automated method can perform as well as dedicated heuristic search methods when it comes to obtaining good differential characteristics. We could find the same differential characteristics as the ones found by designers with our simple framework which incorporates CP. Unlike MILP, CP does not restrict the formulation to linear inequalities or integers. This makes it relatively easier to formulate a cipher and CP's capabilities can be further explored by using various search strategies and solvers available.

Using our tool and its parameterization capabilities, we can easily find differential characteristics that are subjected to some other constraints easily. This allows us to find characteristics to build our distinguishers as well as collision/forgery attacks with just minimal changes to the tool. Our non-black-box limited-birthday distinguishers for Ascon permutation outperform other types of distinguishers for 4, 5 and 6 rounds. With additional constraints added to our Ascon model, we could also find differential characteristics to be used for forgery setting for both reduced-round Ascon-128 and Ascon-128a. We emphasize that our results do not endanger the Ascon design, but they allow to better understand the natural resistance of Ascon against differential cryptanalysis-based attacks.

CP's prowess in searching for differential characteristics is evident in the case of Ascon. In future works, we hope to compare and contrast CP solvers with other automated methods such as MILP/MIP and SAT solvers' capabilities to find differential characteristics on various types of ciphers. This can provide designers and cryptanalysts a good starting point as to what methods to use first in order to find differential characteristics.

Acknowledgements

The authors are grateful to the anonymous reviewers and shepherd for their insightful comments that improved the quality of the paper. The authors are supported by the Temasek Laboratories.

References

[AK18] Ralph Ankele and Stefan Kölbl. Mind the gap-A closer look at the security of block ciphers against differential cryptanalysis. In *International Conference on Selected Areas in Cryptography*, pages 163–190. Springer, 2018.

[AM] Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. https://www.aumasson.jp/data/papers/AM09.pdf.

[BC10a] Christina Boura and Anne Canteaut. A zero-sum property for the KECCAK-f permutation with 18 rounds. In *IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 2488–2492. IEEE, 2010.

[BC10b] Christina Boura and Anne Canteaut. Zero-Sum Distinguishers for Iterated Permutations and Application to Keccak-f and Hamsi-256. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers*, volume 6544 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2010.

[BCC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-Order Differential Properties of Keccak and *Luffa*. In Antoine Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *Lecture Notes in Computer Science*, pages 252–269. Springer, 2011.

[BDK01] Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. *IACR Cryptol. ePrint Arch.*, 2001:21, 2001.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[BM97] Mihir Bellare and Daniele Micciancio. A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 163–192. Springer, 1997.

[BS90] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991.

[CAE19] CAESAR. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. https://competitions.cr.yp.to/caesar.html, 2019.

[CSS+] Geoffrey Chu, Peter J. Stuckey, Andreas Schutt, Thorsten Ehlers, Graeme Gange, and Kathryn Francis. Chuffed, a lazy clause generation solver. https://github.com/chuffed/chuffed.

[DEM15] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR candidates. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 490–509. Springer, 2015.

[DEMS15] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Cryptanalysis of Ascon. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, volume 9048 of *Lecture Notes in Computer Science*, pages 371–387. Springer, 2015.

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

[DEMS21a] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *Journal of Cryptology*, 34(3):33, Jun 2021.

[DEMS21b] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2 Submission to NIST. LWC Final round submission, 2021.

[DGPW12] Alexandre Duc, Jian Guo, Thomas Peyrin, and Lei Wei. Unaligned Rebound Attack: Application to Keccak. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 402–421. Springer, 2012.

[GLMS18] David Gérault, Pascal Lafourcade, Marine Minier, and Christine Solnon. Revisiting AES related-key differential attacks with constraint programming. *Inf. Process. Lett.*, 139:24–29, 2018.

[GLMS20] David Gérault, Pascal Lafourcade, Marine Minier, and Christine Solnon. Computing AES related-key differential characteristics with constraint programming. *Artificial Intelligence*, 278:103183, 2020.

[GMS16] David Gérault, Marine Minier, and Christine Solnon. Constraint Programming Models for Chosen Key Differential Cryptanalysis. In Michel Rueher, editor, *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 584–601. Springer, 2016.

[Goh19] Aron Gohr. Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 150–179. Springer, 2019.

[GP10] Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, volume 6147 of *Lecture Notes in Computer Science*, pages 365–383. Springer, 2010.

[IPS13] Mitsugu Iwamoto, Thomas Peyrin, and Yu Sasaki. Limited-birthday Distinguishers for Hash Functions - Collisions Beyond the Birthday Bound can be Meaningful. *IACR Cryptol. ePrint Arch.*, 2013:611, 2013.

[JNP13] Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Multiple Limited-Birthday Distinguishers and Applications. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 533–550. Springer, 2013.

[Lai94] Xuejia Lai. *Higher Order Derivatives and Differential Cryptanalysis*, pages 227–233. Springer US, Boston, MA, 1994.

[LMS+15] Mario Lamberger, Florian Mendel, Martin Schläffer, Christian Rechberger, and Vincent Rijmen. The Rebound Attack and Subspace Distinguishers: Application to Whirlpool. *J. Cryptol.*, 28(2):257–296, 2015.

[LZWW17] Yanbin Li, Guoyan Zhang, Wei Wang, and Meiqin Wang. Cryptanalysis of round-reduced ASCON. *Sci. China Inf. Sci.*, 60(3):38102, 2017.

[MP13] Nicky Mouha and Bart Preneel. Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Report 2013/328, 2013. https://eprint.iacr.org/2013/328.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.

[NSB+07] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. MiniZinc: Towards a Standard CP Modelling Language. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 529–543. Springer, 2007.

[oST21] National Institute of Standards and Technology. Lightweight cryptography standardization process. https://csrc.nist.gov/projects/lightweight-cryptography, 2021.

[PF] Laurent Perron and Vincent Furnon. Or-tools. https://developers.google.com/optimization/.

[PFL16] Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco Solver Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016.

[RHSS21] Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon. *IACR Trans. Symmetric Cryptol.*, 2021(1):130–155, 2021.

[SGL+17] Siwei Sun, David Gerault, Pascal Lafourcade, Qianqian Yang, Yosuke Todo, Kexin Qiao, and Lei Hu. Analysis of AES, SKINNY, and others with constraint programming. *IACR Transactions on Symmetric Cryptology*, 2017(1):281–306, 2017.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES (L) and other bit-oriented block ciphers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 158–178. Springer, 2014.

[STL19] Christian Schulte, Guido Tack, and Mikael Z. Lagerkvist. Modeling. In Christian Schulte, Guido Tack, and Mikael Z. Lagerkvist, editors, *Modeling and Programming with Gecode*. Gecode, 2019. Corresponds to Gecode 6.2.0.

[Tez16] Cihangir Tezcan. Truncated, Impossible, and Improbable Differential Analysis of ASCON. In Olivier Camp, Steven Furnell, and Paolo Mori, editors, *Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016*, pages 325–332. SciTePress, 2016.

[Tod15] Yosuke Todo. Structural Evaluation by Generalized Integral Property. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314. Springer, 2015.

[Wag99] David A. Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 1999.

[WGR18] Qingju Wang, Lorenzo Grassi, and Christian Rechberger. Zero-Sum Partitions of PHOTON Permutations. In Nigel P. Smart, editor, *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, volume 10808 of *Lecture Notes in Computer Science*, pages 279–299. Springer, 2018.

[ZDW19] Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. Collision attacks on round-reduced gimli-hash/ascon-xof/ascon-hash. *IACR Cryptol. ePrint Arch.*, 2019:1115, 2019.

Appendix

A Example: 5-round non-black-box limited-birthday distinguisher

The 5-round limited-birthday distinguisher is built by extending the 3-round differential characteristic from Table 13 in Appendix B (LB3) forward and backward with probability 1.

Using the technique discussed above, we can obtain the constraints as shown in Table 10. Note that we use $s^n_{r,c}$ to represent the r^{th} row and c^{th} column state value at round n. For this distinguisher, we chose to spend our degrees of freedom at round 1.5, that is after the p_S, but before the p_L of round 1. For each constraint in round 2, we can spend a single bit to fix it. For instance, for the equation $s^2_{2,2} = 0$, if we apply p_L, we have $s^{1.5}_{2,1} \oplus s^{1.5}_{2,2} \oplus s^{1.5}_{2,60} = 0$ (we ignore the effects of p_{AC} here). We can fix one or more of these values to ensure the constraint always holds. For the ones in round 1, we will have to propagate it through p_S. To simplify the computation, we simply fix the active Sboxes values. For instance, we fix the 18^{th} Sbox at round 1, i.e. bits $s^{1.5}_{0,18}, s^{1.5}_{1,18}, s^{1.5}_{2,18}, s^{1.5}_{3,18}, s^{1.5}_{4,18}$ such that $s^1_{1,18} = 0$ and $s^1_{3,18} = s^1_{4,18} \oplus 1$ are satisfied.

Table 10: Constraints generated by active Sboxes of LB3.

round	**constraints**		
0	$s^0_{0,63} = s^0_{4,63} \oplus 1$	$s^0_{1,63} = s^0_{2,63} \oplus 1$	
1	$s^1_{1,18} = 0$ $s^1_{1,27} = 0$ $s^1_{1,63} = 0$	$s^1_{3,18} = s^1_{4,18} \oplus 1$ $s^1_{3,27} = s^1_{4,27} \oplus 1$ $s^1_{3,63} = s^1_{4,63} \oplus 1$	
2	$s^2_{2,2} = 0$	$s^2_{3,2} = 1$	$s^2_{0,2} = s^2_{4,2} \oplus 1$
	$s^2_{2,15} = 0$	$s^2_{3,15} = 1$	$s^2_{0,15} = s^2_{4,15} \oplus 1$
	$s^2_{2,18} = 0$	$s^2_{3,18} = 1$	$s^2_{0,18} = s^2_{4,18} \oplus 1$
	$s^2_{2,24} = 0$	$s^2_{3,24} = 1$	$s^2_{0,24} = s^2_{4,24} \oplus 1$
	$s^2_{2,27} = 0$	$s^2_{3,27} = 1$	$s^2_{0,27} = s^2_{4,27} \oplus 1$
	$s^2_{1,37} = 0$		$s^2_{3,37} = s^2_{4,37} \oplus 1$
	$s^2_{2,38} = 0$	$s^2_{3,38} = 1$	$s^2_{0,38} = s^2_{4,38} \oplus 1$
	$s^2_{1,55} = 0$		$s^2_{3,55} = s^2_{4,55} \oplus 1$
	$s^2_{2,57} = 0$	$s^2_{3,57} = 1$	$s^2_{0,57} = s^2_{4,57} \oplus 1$
	$s^2_{2,60} = 0$	$s^2_{3,60} = 1$	$s^2_{0,60} = s^2_{4,60} \oplus 1$
	$s^2_{2,63} = 1$	$s^2_{3,63} = 1$	
	$s^2_{4,63} = 1$		$s^2_{0,63} = s^2_{1,63} \oplus 1$

For the 2 constraints in the 63^{rd} Sbox in round 0, we can substitute them into the ANF of the Sbox. We obtain the following equations:

$$y_0 = x_2 \oplus x_3 \oplus x_4 \oplus 1 \qquad y_1 = 0 \qquad y_2 = x_3x_4 \oplus x_4$$
$$y_3 = x_3x_4 \qquad y_4 = x_3 \oplus x_4$$

To maintain everything linear (with respect to the output of the Sbox), we only kept the second equation ($y_1 = 0$) i.e. we have $s^{0.5}_{1,63} = 0$ and ignore the other independent equation that is nonlinear. Since the backward diffusion is strong in Ascon permutation, a total of 33 different Sboxes at round 1 are affecting this constraint. Table 11 shows all the bits/Sboxes involved for a particular constraint. Note that the bits involved for each constraint as shown are linearly related to the constraint. For instance, changing the parity of $s^{1.5}_{0,2}$ changes the parity of the constraint $s^2_{0,2} = s^2_{4,2} \oplus 1$. Similarly, for those represented using Sboxes, changing the r^{th} bit of the input of an Sbox changes the parity of the r^{th} bit in the same Sbox as the constraint. The bits that are in red are the bits we can use to fix the particular constraint. Note that this selection is not necessarily unique. All in all, we can fix a total of 39 constraints and thus the probability is 2^{-1}, while the distinguisher complexity is 2^2 permutation calls.

Table 11: This table shows the bits involved for each constraint in LB3. $s^n_{r,c}$ represents the r^{th} row and c^{th} column state value at round n. S^n_v represents all the bits in the v^{th} Sbox at round n. Note that the bits highlighted in red are the bits we use to fix the particular constraint.

constraint	**bits involved**	**constraint**	**bits involved**
$s^1_{1,18} = 0$	${\color{red}S^{1.5}_{18}}$	$s^2_{2,2} = 0$	${\color{red}s^{1.5}_{2,2}}, s^{1.5}_{2,1}, s^{1.5}_{2,60}$
$s^1_{3,18} = s^1_{4,18} \oplus 1$		$s^2_{3,2} = 1$	${\color{red}s^{1.5}_{3,2}}, s^{1.5}_{3,56}, s^{1.5}_{3,49}$
$s^1_{1,27} = 0$	${\color{red}S^{1.5}_{27}}$	$s^2_{2,15} = 0$	${\color{red}s^{1.5}_{2,15}}, s^{1.5}_{2,14}, s^{1.5}_{2,9}$
$s^1_{3,27} = s^1_{4,27} \oplus 1$		$s^2_{3,15} = 1$	${\color{red}s^{1.5}_{3,15}}, s^{1.5}_{3,5}, s^{1.5}_{3,62}$
$s^1_{1,63} = 0$	${\color{red}S^{1.5}_{63}}$	$s^2_{2,18} = 0$	$s^{1.5}_{2,18}, {\color{red}s^{1.5}_{2,17}}, s^{1.5}_{2,12}$
$s^1_{3,63} = s^1_{4,63} \oplus 1$		$s^2_{3,18} = 1$	$s^{1.5}_{3,18}, {\color{red}s^{1.5}_{3,8}}, s^{1.5}_{3,1}$
$s^2_{0,2} = s^2_{4,2} \oplus 1$	${\color{red}s^{1.5}_{0,2}}, s^{1.5}_{0,47}, s^{1.5}_{0,38}, s^{1.5}_{4,2}, s^{1.5}_{4,59}, s^{1.5}_{4,25}$	$s^2_{2,24} = 0$	${\color{red}s^{1.5}_{2,24}}, s^{1.5}_{2,23}, s^{1.5}_{2,18}$
$s^2_{0,15} = s^2_{4,15} \oplus 1$	${\color{red}s^{1.5}_{0,15}}, s^{1.5}_{0,60}, s^{1.5}_{0,51}, s^{1.5}_{4,15}, s^{1.5}_{4,08}, s^{1.5}_{4,38}$	$s^2_{3,24} = 1$	${\color{red}s^{1.5}_{3,24}}, s^{1.5}_{3,14}, s^{1.5}_{3,7}$
$s^2_{0,18} = s^2_{4,18} \oplus 1$	$s^{1.5}_{0,18}, s^{1.5}_{0,63}, {\color{red}s^{1.5}_{0,54}}, s^{1.5}_{4,18}, s^{1.5}_{4,11}, s^{1.5}_{4,41}$	$s^2_{2,27} = 0$	$s^{1.5}_{2,27}, {\color{red}s^{1.5}_{2,26}}, s^{1.5}_{2,21}$
$s^2_{0,24} = s^2_{4,24} \oplus 1$	${\color{red}s^{1.5}_{0,24}}, s^{1.5}_{0,5}, s^{1.5}_{0,60}, s^{1.5}_{4,24}, s^{1.5}_{4,17}, s^{1.5}_{4,47}$	$s^2_{3,27} = 1$	$s^{1.5}_{3,27}, {\color{red}s^{1.5}_{3,17}}, s^{1.5}_{3,10}$
$s^2_{0,27} = s^2_{4,27} \oplus 1$	$s^{1.5}_{0,27}, {\color{red}s^{1.5}_{0,8}}, s^{1.5}_{0,63}, s^{1.5}_{4,27}, s^{1.5}_{4,20}, s^{1.5}_{4,50}$	$s^2_{1,37} = 0$	${\color{red}s^{1.5}_{1,37}}, s^{1.5}_{1,40}, s^{1.5}_{1,62}$
$s^2_{3,37} = s^2_{4,37} \oplus 1$	${\color{red}s^{1.5}_{3,37}}, s^{1.5}_{3,27}, s^{1.5}_{3,20}, s^{1.5}_{4,37}, s^{1.5}_{4,30}, s^{1.5}_{4,60}$	$s^2_{2,38} = 0$	${\color{red}s^{1.5}_{2,38}}, s^{1.5}_{2,37}, s^{1.5}_{2,32}$
$s^2_{0,38} = s^2_{4,38} \oplus 1$	${\color{red}s^{1.5}_{0,38}}, s^{1.5}_{0,19}, s^{1.5}_{0,10}, s^{1.5}_{4,38}, s^{1.5}_{4,31}, s^{1.5}_{4,61}$	$s^2_{3,38} = 1$	${\color{red}s^{1.5}_{3,38}}, s^{1.5}_{3,28}, s^{1.5}_{3,21}$
$s^2_{3,55} = s^2_{4,55} \oplus 1$	${\color{red}s^{1.5}_{3,55}}, s^{1.5}_{3,45}, s^{1.5}_{3,38}, s^{1.5}_{4,55}, s^{1.5}_{4,48}, s^{1.5}_{4,14}$	$s^2_{1,55} = 0$	${\color{red}s^{1.5}_{1,55}}, s^{1.5}_{1,58}, s^{1.5}_{1,16}$
$s^2_{0,57} = s^2_{4,57} \oplus 1$	${\color{red}s^{1.5}_{0,57}}, s^{1.5}_{0,38}, s^{1.5}_{0,29}, s^{1.5}_{4,57}, s^{1.5}_{4,50}, s^{1.5}_{4,16}$	$s^2_{2,57} = 0$	${\color{red}s^{1.5}_{2,57}}, s^{1.5}_{2,56}, s^{1.5}_{2,51}$
$s^2_{0,60} = s^2_{4,60} \oplus 1$	${\color{red}s^{1.5}_{0,60}}, s^{1.5}_{0,41}, s^{1.5}_{0,32}, s^{1.5}_{4,60}, s^{1.5}_{4,53}, s^{1.5}_{4,19}$	$s^2_{3,57} = 1$	${\color{red}s^{1.5}_{3,57}}, s^{1.5}_{3,47}, s^{1.5}_{3,40}$
$s^2_{0,63} = s^2_{1,63} \oplus 1$	$s^{1.5}_{0,63}, {\color{red}s^{1.5}_{0,44}}, s^{1.5}_{0,35}, s^{1.5}_{1,63}, s^{1.5}_{1,2}, s^{1.5}_{1,24}$	$s^2_{2,60} = 0$	${\color{red}s^{1.5}_{2,60}}, s^{1.5}_{2,59}, s^{1.5}_{2,54}$
$s^2_{3,60} = 1$	${\color{red}s^{1.5}_{3,60}}, s^{1.5}_{3,50}, s^{1.5}_{3,43}$	$s^2_{2,63} = 1$	$s^{1.5}_{2,63}, {\color{red}s^{1.5}_{2,62}}, s^{1.5}_{2,57}$
$s^2_{3,63} = 1$	$s^{1.5}_{3,63}, {\color{red}s^{1.5}_{3,53}}, s^{1.5}_{3,46}$	$s^2_{4,63} = 1$	$s^{1.5}_{4,63}, {\color{red}s^{1.5}_{4,56}}, s^{1.5}_{4,22}$
$s^{0.5}_{1,63} = 0$	$S^{1.5}_{63}, S^{1.5}_{62}, S^{1.5}_{61}, S^{1.5}_{60}, S^{1.5}_{59}, S^{1.5}_{55}, {\color{red}S^{1.5}_{52}}, S^{1.5}_{50}, S^{1.5}_{49}, S^{1.5}_{47}, S^{1.5}_{44}, S^{1.5}_{42}$ $S^{1.5}_{40}, S^{1.5}_{39}, S^{1.5}_{38}, S^{1.5}_{36}, S^{1.5}_{35}, S^{1.5}_{34}, S^{1.5}_{33}, S^{1.5}_{28}, S^{1.5}_{24}, S^{1.5}_{20}, S^{1.5}_{19}, S^{1.5}_{18}$ $S^{1.5}_{16}, S^{1.5}_{15}, S^{1.5}_{12}, S^{1.5}_{10}, S^{1.5}_{9}, S^{1.5}_{8}, S^{1.5}_{6}, S^{1.5}_{3}, S^{1.5}_{2}$		

Computing the generic cost. We extend LB3 backward and forward with probability 1 for one round (i.e. for each active Sbox, we include all possible differences into the set):

$$
\begin{aligned}
D_{in} &= \{\Delta_{in} \text{ s.t. } \mathrm{DDT}(\Delta_{in} \to p_L^{-1}(LB3[0])) > 0\} \\
D_{out} &= \{\Delta_{out} \text{ s.t. } \mathrm{DDT}(LB3[3] \to \Delta_{out}) > 0\}
\end{aligned}
$$

Where LB3[0] and LB3[3] refer to the start of the first round and end of the last round of LB3 respectively. As an example, we calculate:

$$\begin{aligned}|D_{out}| =& (|\mathrm{DDT}[\texttt{0x02},*] > 0|)^8 \cdot (|\mathrm{DDT}[\texttt{0x04},*] > 0|)^9 \cdot (|\mathrm{DDT}[\texttt{0x06},*] > 0|)^{10} \cdot \\ & (|\mathrm{DDT}[\texttt{0x08},*] > 0|) \cdot (|\mathrm{DDT}[\texttt{0x0a},*] > 0|)^2 \cdot (|\mathrm{DDT}[\texttt{0x0c},*] > 0|) \cdot \\ & (|\mathrm{DDT}[\texttt{0x10},*] > 0|)^2 \cdot (|\mathrm{DDT}[\texttt{0x12},*] > 0|) \cdot (|\mathrm{DDT}[\texttt{0x16},*] > 0|) \cdot \\ & (|\mathrm{DDT}[\texttt{0x1a},*] > 0|)^2 \\ =& 2^{115}\end{aligned}$$

The generic complexity to generate the same limited-birthday property for a random permutation is

$$C(|D_{in}|, |D_{out}|) = \max\{\min\{2^{75.51}, 2^{103}\}, 2^{36.02}\} = 2^{75.51}$$

B Limited-birthday distinguishers

Table 12: Differential characteristic LB2. The probability of this characteristic is 2^{-8}. The breakdown of the probability (in $-\log_2$) is $[2, 6]$.

	input difference	after 1 round	after 2 rounds
x_0	0000000000000000	0000201000000001	0000000004000101
x_1	0000000000000001	0000000000000000	2001209002000049
x_2	0000000000000001	0000000000000000	0000000000000000
x_3	0000000000000000	0000000000000000	0000000000000000
x_4	0000000000000000	0000000000000000	0000000000000000

Table 13: 3-round differential characteristic for 5-round non-black-box limited-birthday distinguisher (4 round on black-box) on the ASCON permutation. The differential probability is 2^{-40}. The breakdown of the probability (in $-\log_2$) is $[2, 6, 32]$. Fixing the state at the start of round 3, we can achieve the distinguisher with a complexity of 2^2 permutation calls.

	input difference	after 1 round	after 2 rounds	after 3 rounds
r_0	0000000000000000	0000201000000001	0000000004000101	4020100004000180
r_1	0000000000000001	0000000000000000	2001209002000049	0008000224000900
r_2	0000000000000001	0000000000000000	0000000000000000	9481b45a4308006c
r_3	0000000000000000	0000000000000000	0000000000000000	322d30d8b6488148
r_4	0000000000000000	0000000000000000	0000000000000000	0000000000000000

Table 14: Differential characteristic LB3.1. The probability of this characteristic is 2^{-8}. The breakdown of the probability (in $-\log_2$) is $[47, 12, 6]$.

	input difference	after 1 round	after 2 rounds	after 3 rounds
r_0	32a11104c9b008db	0000201000000001	0000000004000101	4020301004000181
r_1	0000000000000001	0000000000000000	0000000000000000	0008000226000909
r_2	0000000000000001	0000000000000000	0000000000000000	0000000000000000
r_3	32a11104c9b008da	0000201000000001	0000000000000000	0000000000000000
r_4	32a11104c9b008da	0000000000000000	0000000000000000	0000000000000000

Table 15: Differential characteristic LB4. The probability of this characteristic is 2^{-147}. The breakdown of the probability (in $-\log_2$) is $[2, 6, 32, 107]$.

	input difference	**after 1 round**	**after 2 rounds**	**after 3 rounds**	**after 4 rounds**
x_0	0000000000000000	0000201000000001	0000000004000101	0020100000000100	162e14c670b19a21
x_1	0000000000000001	0000000000000000	2001209002000049	2009241226000948	0012000210000d48
x_2	0000000000000001	0000000000000000	0000000000000000	9481b45a4308006c	645f5698151c0c77
x_3	0000000000000000	0000000000000000	0000000000000000	322d30d8b6488148	99b7ea6001186aa2
x_4	0000000000000000	0000000000000000	0000000000000000	1002000000080008	6648288901610300

Table 16: Differential characteristic LB4.1. The probability of this characteristic is 2^{-109}. The breakdown of the probability (in $-\log_2$) is $[58, 12, 9, 30]$.

	input difference	**after 1 round**	**after 2 rounds**	**after 3 rounds**	**after 4 rounds**
x_0	0000000400000000	0000000000000000	0000000000000000	080420140000c041	98100d240cc44291
x_1	63b6c53b00766181	0000000401020000	0000000000000000	0000000000000000	283420b1cc948e80
x_2	63b6c53b00766181	0000000000000000	0000000000000000	0000000000000000	0000000000000000
x_3	0000000400000000	0000000401020000	0000000400004001	0000000000000000	0000000000000000
x_4	0000000400000000	0000000000000000	0000000000000000	0000002408804081	0000002408804081

Table 17: Differential characteristic LB4.2. The probability of this characteristic is 2^{-141}. The breakdown of the probability (in $-\log_2$) is $[82, 32, 4, 23]$.

	input difference	after 1 round	after 2 rounds
x_0	fb8e401124ca8085	4020001000000100	2000000000000001
x_1	04318d0c40007a10	0020300000000181	0000000000000000
x_2	04318d0c40007a10	0020300000000001	0000000000000000
x_3	fb8c400120408005	4000001004010000	0000000000000000
x_4	fb8c400120408005	0000000004010181	0000000000000000

	after 3 rounds	after 4 rounds
x_0	2000241200000001	0000040204800121
x_1	2000000002400008	2401048202000041
x_2	0000000000000000	108000000369000c
x_3	0000000000000000	0204000002409128
x_4	0000000000000000	0000000000000000

Table 18: Differential characteristic LB5. The probability of this characteristic is 2^{-237}. The breakdown of the probability (in $-\log_2$) is $[6, 9, 30, 81, 111]$.

	input difference	after 1 round	after 2 rounds
x_0	0000000000020081	0000000000000000	60208402100a0000
x_1	0000000000000000	0000000000000000	0000000000000000
x_2	0000000000000000	0000000000000000	0000000000000000
x_3	0000000000020081	2000800000020000	0000000000000000
x_4	0000000000020081	0000000000000000	2040800000120440

	after 3 rounds	after 4 rounds	after 5 rounds
x_0	61e8c00a141a0442	644998a100440322	83d466293fa88565
x_1	4740141a1058e64a	0000100800482400	0948c84107473492
x_2	0000000000184000	44241d484669b184	579146a2e5018394
x_3	0000000000000000	e42585812e40b044	c3220c515630a665
x_4	61e8c00a141a0442	e5619ca12420a2a4	5041813b7a143040

C Conforming pairs for distinguishers

Table 19: Conforming pairs for limited-birthday distinguishers.

# Characteristic	pair 1	pair 2
LB2	0ec9a62c2c63f2e1 79f76978592b5f9a 8b7e1c6f36e61951 7892d3bc3ff6675f ec501e18515c2b2a	4ced65d55729c68a 8c98d0e46b567147 ad37b7a235df2793 0cf732cc56f57320 0d5beee4185b1940
LB3	a91800bccb3e1021 ae0b695f555f01b6 6da3c62ed382a546 935d5c6d4457e8a6 9ab5e27340f39ef3	ff3eebddd0583072 4f6212c337663f58 da8cecd388d99fa6 9537c62c3718f8df 1dbf6322518c9aa6
LB4	aa56bfb76c5c2d61 16f71384e9511df8 08e3f0011908e3a9 f16e9f3e53623a26 644bf9d542b58791	cd7e574e14073399 a5b0a848ea6c3377 e9accb5d7325cb68 e708cd5f2b2c2859 436fb2383be7b3e0
LB4.2	71aa37de1c1dd67b 6c370ee346a8dc96 dd5f2e21b9b4dc62 4397c4c71205ed80 bd3bea710986ad09	8a2477cf38d756fe 680683ef06a8a686 d96ea32df9b4a672 b81b84c632456d85 46b7aa7029c62d0c

Table 20: Conforming pairs for 4 rounds of rectangle distinguishers.

pair 1-1	pair 1-2
bf6c940a612235b2 9dd38c49d55b7149 2ec56c0721ffa7b2 36731ba9da4b939d b2c7651aaf45b4f4	bf6c940a612235b2 9dd38c49555b7149 2ec56c07a1ffa7b2 36731ba9da4b939d b2c7651aaf45b4f4
pair 2-1	**pair 2-2**
fb93bc942f3c9bbf 6eb83e2af39c75d2 fc14891d4b11709e 38393fe9df45af24 2a8d827dcb01dcd0	fb93bc942f3c9bbf 6eb83e2a739c75d2 fc14891dcb11709e 38393fe9df45af24 2a8d827dcb01dcd0

D Forgery characteristics

Table 21: Differential characteristic to create forgery for round-reduced Ascon-128a with a 3-round finalization. The differential probability is 2^{-19}. The breakdown of the probability (in $-\log_2$) is $[4, 6, 9]$.

	input difference	after 1 round	after 2 rounds	after 3 rounds
r_0	0000000000000001	0000000000000000	0000000000000000	????????????????
r_1	0000000000000001	0000000000000000	0000000000000000	????????????????
r_2	0000000000000000	8400000000000001	4010000000000001	????????????????
r_3	0000000000000000	0000000000000000	8461c20000000001	4000240800000000
r_4	0000000000000000	0000000000000000	0000000000000000	8769018400c230e0

Table 22: Differential characteristic to create forgery for round-reduced Ascon-128a with a 3-round permutation. The differential probability is 2^{-116}. The breakdown of the probability (in $-\log_2$) is $[8, 44, 64]$

	input difference	after 1 round	after 2 rounds	after 3 rounds
r_0	0040000400001004	0000000000000000	2041800c26009004	f00e0594e37e2707
r_1	0000000000000000	0a40000408001024	0210800812052004	2e250895044c9a03
r_2	0000000000000000	0000000000000000	0000000000000000	0000000000000000
r_3	0000000000000000	0000000000000000	0000000000000000	0000000000000000
r_4	0000000000000000	0a40800c0a003024	0614811812052004	0000000000000000

Table 23: Differential characteristic to create forgery for round-reduced Ascon-128 and Ascon-128a with a 3-round finalization. The differential probability is 2^{-32}. The breakdown of the probability (in $-\log_2$) is $[2, 16, 14]$

	input difference	after 1 round	after 2 rounds	after 3 rounds
r_0	0000000000000001	0000000000000000	0000000000000000	????????????????
r_1	0000000000000000	0000000002000009	1201000004840000	????????????????
r_2	0000000000000000	0000000000000000	a40000000308000d	????????????????
r_3	0000000000000000	0000000000000000	0204000002008108	b76e5b40850d4183
r_4	0000000000000000	0200000000800001	0204400000800000	100100408604800a

Table 24: Differential characteristic to create forgery for round-reduced Ascon-128 with a 4-round finalization. The differential probability is 2^{-100}. The breakdown of the probability (in $-\log_2$) is $[2, 14, 50, 34]$

	input difference	**after 1 round**	**after 2 rounds**
r_0	0000000000000001	0000201000000001	0000201004000100
r_1	0000000000000000	0000000002000009	2005209002000008
r_2	0000000000000000	0000000000000000	a40000000308000d
r_3	0000000000000000	0000000000000000	0204000002008108
r_4	0000000000000000	0000000000000000	0000000000000000

	after 3 rounds	**after 4 rounds**
r_0	1208004020008000	????????????????
r_1	0008041024400000	????????????????
r_2	7011b45a4280200b	????????????????
r_3	876642c2a5494081	720d846c9c95a340
r_4	0206004084098002	c76a026745c07121

E 5-round boomerang characteristic

Table 25: Boomerang characteristic for 5 rounds of the Ascon permutation. The probability of the boomerang characteristic is 2^{-96} but the probability of the rectangle characteristic is $2^{-85.57}$. The breakdown of the probability (in $-\log_2$) for the upper and lower characteristics are $[2, 6]$ and $[2, 6, 32]$ respectively

		input difference	**after 1 round**	**after 2 rounds**
Upper characteristic	r_0	0000000000000000	0201000000001000	0000004000101000
	r_1	0000000000001000	0000000000000000	1209002000049200
	r_2	0000000000001000	0000000000000000	0000000000000000
	r_3	0000000000000000	0000000000000000	0000000000000000
	r_4	0000000000000000	0000000000000000	0000000000000000

		input difference	**after 1 round**	**after 2 rounds**	**after 3 rounds**
Lower characteristic	r_0	0000000000000000	0000201000000001	0000000004000101	4020100004000180
	r_1	0000000000000001	0000000000000000	2001209002000049	0008000224000900
	r_2	0000000000000001	0000000000000000	0000000000000000	9481b45a4308006c
	r_3	0000000000000000	0000000000000000	0000000000000000	322d30d8b6488148
	r_4	0000000000000000	0000000000000000	0000000000000000	0000000000000000

F 2-round differential characteristic for ASCON-HASH

Table 26: Differential characteristic for 2 rounds of ASCON permutation. The probability of this characteristic is 2^{-156}. The breakdown of the probability (in $-\log_2$) is $[54, 102]$

	input difference	after 1 round	after 2 rounds
r_0	bb450325d90b1581	2201080000011080	baf571d85e1153d7
r_1	0000000000000000	2adf0c201225338a	0000000000000000
r_2	0000000000000000	0000000000000000	0000000000000000
r_3	0000000000000000	0000000100408000	0000000000000000
r_4	0000000000000000	2adf0c211265b38a	0000000000000000