Efficient Methods to Search for Best Differential Characteristics on SKINNY

Stéphanie Delaune[1(✉)], Patrick Derbez[1], Paul Huynh[2], Marine Minier[2], Victor Mollimard[1], and Charles Prud'homme[3]

[1] Univ. Rennes, CNRS, IRISA, Rennes, France
{stephanie.delaune,patrick.derbez,victor.mollimard}@irisa.fr
[2] Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France
{paul.hyunh,marine.minier}@loria.fr
[3] IMT-Atlantique, TASC, LS2N, Nantes, France
charles.prudhomme@imt-atlantique.fr

Abstract. Evaluating resistance of ciphers against differential cryptanalysis is essential to define the number of rounds of new designs and to mount attacks derived from differential cryptanalysis.

In this paper, we propose automatic tools to find the best differential characteristics on the SKINNY block cipher. As usually done in the literature, we split this search in two stages denoted by Step 1 and Step 2. In Step 1, we aim at finding all truncated differential characteristics with a low enough number of active Sboxes. Then, in Step 2, we try to instantiate each difference value while maximizing the overall differential characteristic probability. We solve Step 1 using an ad-hoc method inspired from the work of Fouque *et al.* whereas Step 2 is modelized for the Choco-solver library as it seems to outperform all previous methods on this stage.

Notably, for SKINNY-128 in the **SK** model and for 13 rounds, we retrieve the results of Abdelkhalek *et al.* within a few seconds (to compare with 16 days) and we provide, for the first time, the best differential related-tweakey characteristics up to 14 rounds for the **TK1** model. Regarding the **TK2** and the **TK3** models, we were not able to test all the solutions Step 1, and thus the differential characteristics we found up to 16 and 17 rounds are not necessarily optimal.

Keywords: Differential cryptanalysis · Automatic tools · SKINNY

1 Introduction

Differential cryptanalysis [5] evaluates the propagation of an input difference $\delta X = X \oplus X'$ between two plaintexts X and X' through the ciphering process. Indeed, differential attacks exploit the fact that the probability of observing a

The research leading to these results has received funding from the French National Research Agency (ANR) under the project Decrypt ANR-18-CE39-0007.

K. Sako and N. O. Tippenhauer (Eds.): ACNS 2021, LNCS 12727, pp. 184–207, 2021.
https://doi.org/10.1007/978-3-030-78375-4_8

specific output difference given a specific input difference is not uniformly distributed. Today, differential cryptanalysis is public knowledge, and block ciphers such as AES have proven bounds against differential attacks. A classical extension of differential cryptanalysis is the so called related-key differential cryptanalysis [4] that allows an attacker to inject differences not only between the plaintexts X and X' but also between the keys K and K' (even if the secret key K stays unknown from the attacker). This attack has been recently extended to tweakable block ciphers [3]. Those particular ciphers allow in addition to the key, a public value called a tweak. Thus, related-tweakey differential attacks allow related-key differences but also related-tweak differences (*i.e.* differences in a pair of tweaks (T, T')). In differential attacks, two notions are considered: first, differentials where only the input and the output differences are known; and differential characteristics where each difference after each round is completely specified. A classical approach to evaluate the resistance against differential attacks is to compute the probability of the best differential characteristic of the cipher.

Finding optimal (related-tweakey) differential characteristics is a highly combinatorial problem that hardly scales. To limit this explosion, a common solution consists in using a truncated representation [16] for which cells are abstracted by single bits that indicate whether sequences contain differences or not. Typically, each cell (*i.e.* byte or nibble) is abstracted by a single bit (or, equivalently, a Boolean value). In this case, the goal is no longer to find the exact input and output differences, but to find the positions of these differences, *i.e.*, the presence or absence of a difference for every cell. When a difference is present at the input of an S-box, we talk about an active S-box or an active byte/nibble. However, some truncated representations may not be valid (*i.e.*, there do not exist actual byte values corresponding to these difference positions) because some constraints at the byte level are relaxed when reasoning on difference positions.

Hence, the optimal (related-tweakey) differential characteristic problem is usually solved in two steps [1,6]. In the first one, every differential byte is abstracted by a Boolean variable, denoted by Δ, that indicates whether there is a difference or not at this position, and we search for all truncated representations of low weight as the less differences passing through S-boxes there are, the more the probability is increased. Then, for each of these low weight truncated representations, the second step aims at deciding whether it is valid (*i.e.*, whether it is possible to find actual cell values, denoted δ, for every Boolean variable) and, if it is valid, at finding the actual cell values that maximize the probability of obtaining the output difference given the input difference.

Related Work. Many techniques have been proposed to search for the Step 1 solutions using automatic tools such as Boolean satisfiability (SAT) [21,26,27] or Mixed Integer Linear Programming (MILP) [3,24,30] and Satisfiability Modulo Theories (SMT) [17]. Dedicated solutions have also been proposed [20].

Regarding the search of the best instantiation of a truncated characteristic, most of the approaches were ad-hoc and dedicated to a precise cipher [6,9–11,18,28]. Concerning the use of SAT solvers, [28] implements a SAT model for

differential cryptanalysis based on `Cryptominisat5` [26] for `Midori64` and `LED64`. This model implies a sufficiently small number of clauses to model the non-zero values of the `DDT` and to be applicable. However, no result concerning 8-bit S-boxes are given. As SAT uses Boolean formulas, it seems that the same problem than for MILP appears for modeling S-box: a huge number of Boolean formulas will be necessary to correctly model this step even if dedicated tools as Logic Friday or the Expresso algorithm [1] are used. In [1], 16 days are needed to find the best related tweakey differential characteristics on `SKINNY`-128 for the **SK** model. Recently, in [11,12], the authors introduce Constraint Programming (CP) models for Step 2 and the performance results are really promising regarding AES-192 and AES-256.

Our Contribution. In this paper, we refine the security bounds on the `SKINNY`-n tweakable block cipher regarding differential cryptanalysis for the four following attack models according to the size of the tweakey: the **SK** model focuses on single-key attack, the **TK1** model considers related-tweakey attack when the tweakey has only one component, the **TK2** model in the related-tweakey settings considers 2 components and the **TK3** model, 3 components.

To do so, we implement Step 1 using an ad-hoc method inspired from [10]. We also propose a CP model for Step 2 taking as input the solutions outputted by Step 1. Thus, we provide, for the first time, the best differential related-tweakey characteristics up to 14 rounds for the **TK1** model. We also consider the **TK2** and **TK3** models and we were able to found some differential characteristics up to 16 rounds for the **TK2** model and up to 17 rounds for the **TK3** model of `SKINNY`-128. However, we were not able to test all the solutions Step 1, and thus these differential characteristics are not necessarily optimal. This is an important improvement compared to previous results. For instance, in [19] Liu *et al.* could only find the best differential characteristics up to 7 and 9 rounds for **TK1** and **TK2**. Finally we also show there is no differential characteristic with probability higher than 2^{-128} against 15 rounds in the **TK1** model, 19 rounds for **TK2** and 23 rounds for **TK3**. All those results clearly show that `SKINNY` is much more resistant to differential cryptanalysis than one would expect while counting the number of active S-boxes.

As a feedback, we also provide the time results we obtain when implementing the Step 1 using another tool, a MILP model for the 4 attack settings. As a result we show that MILP is not always the best choice. First, for Step 1, the ad-hoc method is able to surpass the MILP model. Second, the CP model proposed for Step 2 is incomparably much faster than the MILP model proposed in [1] that requires 16 days according their paper.

All the codes to reproduce these results can be found at [7].

Organization of the Paper. Section 2 gives a short description of `SKINNY`-n; Sect. 3 presents our Ad-Hoc tool and gives performance results comparing our Ad-Hoc model with a MILP one; Sect. 4 presents our dedicated modeling for Step 2 based on CP and analyzes the obtained results. Finally, Sect. 5 concludes this paper.

2 Cipher Under Study: SKINNY-n

In this section, we briefly review the tweakable block cipher SKINNY-n where n denotes the block size and can be equal to 64 or 128 bits. All the details that have been overlooked can be found in [3].

As its name indicates, it enciphers blocks of length 64 or 128 bits seen as a 4×4 matrix of cells (nibbles for $n = 64$ or bytes for $n = 128$). We denote $x_{i,j,k}$ the cell at row i and column j of the internal state at the beginning of round k (i.e. $0 \leq i, j \leq 3$ and $0 \leq k \leq r + 1$ where r is the number of rounds depending on the tweak length and on the key length). SKINNY-n follows the TWEAKEY framework from [15]. SKINNY-n has three main tweakey size versions: the tweakey size can be equal to $t = 64$ or 128 bits, $t = 128$ or 256 bits and $t = 192$ or 384 bits and we denote $z = t/n$ the tweakey size to block size ratio. Then, the number of rounds is directly derived from the z value: between 32 rounds for the 64/64 version up to 56 for the 128/384 version.

The tweakey state is also viewed as a collection of z 4×4 square arrays of cells (nibbles for $n = 64$ or bytes for $n = 128$). We denote these arrays $TK1$ when $z = 1$, $TK1$ and $TK2$ when $z = 2$, and finally $TK1$, $TK2$ and $TK3$ when $z = 3$. We also denote by $TKk_{i,j}$ the nibble or the byte at position $[i, j]$ in TKk. Moreover, we define the associated adversarial model **SK** (resp. **TK1**, **TK2** or **TK3**) where the attacker cannot (resp. can) introduce differences in the tweakey state.

One encryption round of SKINNY is composed of five operations applied in the following order: SubCells (SC), AddConstants (AC), AddRoundTweakey (ART), ShiftRows (SR) and MixColumns (MC) (see Fig. 1).

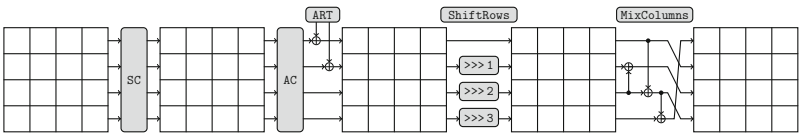

Fig. 1. The SKINNY round function with its five transformations [14].

SubCells. A 4-bit ($n = 64$) or an 8-bit ($n = 128$) S-box is applied to each cell of the state. See [3] for the details of the S-boxes.

AddConstants. A 6-bit affine LFSR is used to generate round constants c_0 and c_1 that are XORed to the state at position $[0, 0]$ and $[1, 0]$ whereas the constant $c_2 =$ 0x02 is XORed to the position $[2, 0]$.

AddRoundTweakey. The first and second rows of all tweakey arrays are extracted and bitwise exclusive-ored to the cipher internal state, respecting the array positioning. More formally, we have:

- $x_{i,j} = x_{i,j} \oplus TK1_{i,j}$ when $z = 1$,
- $x_{i,j} = x_{i,j} \oplus TK1_{i,j} \oplus TK2_{i,j}$ when $z = 2$,

– $x_{i,j} = x_{i,j} \oplus TK1_{i,j} \oplus TK2_{i,j} \oplus TK3_{i,j}$ when $z = 3$.

Then, the tweakey arrays are updated. First, a permutation P_T is applied on the cells positions of all tweakey arrays: if $\ell = 4 * i + j$ where i is the row index and j is the column index, then the cell ℓ is moved to position $P_T(\ell)$ where $P_T = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$. Second, every cell of the first and second rows of $TK2$ and $TK3$ are individually updated with an LFSR on 4 bits (when $n = 64$) or on 8 bits (when $n = 128$) with a period equal to 15.

`ShiftRows.` The rows of the cipher state cell array are rotated to the right. More precisely, the second (resp. third and fourth) cell row is rotated by 1 position (resp. 2 and 3 positions).

`MixColumns.` Each column of the cipher internal state array is multiplied by the 4×4 binary matrix M:

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

Since 2016 and the birth of `SKINNY`-128, the cryptographic world never stopped trying to attack it. Among all the cryptanalysis results, we could cite the following ones in the related-tweakey settings and classified according the type of attacks. First, in [19,25,31], boomerang and rectangle related-tweakey attacks are considered. The best result is on 28 rounds with a complexity of 2^{315} in time based on a boomerang distinguisher of 23 rounds in the **TK3** scenario. Concerning impossible related-tweakey attack [19,29], the best attack has 23 rounds using a distinguisher with 15 rounds in the **TK2** scenario. Even if the distinguishers presented here have less rounds, they do not look at the same attack scenario. This paper essentially goes further than [1] concerning the search of the best related-tweakey differential trails and aims at refining the best security bounds of `SKINNY` in this attack model.

3 Models and Results for Step 1

As explained in the introduction, in a first step called Step 1, we abstract each possible difference at cell (nibble or byte) level by a binary variable which symbolizes the presence/absence of a difference value at a given position of the cipher. The main concern regarding this step is the combinatorial explosion induced by the abstract XOR operation for which the sum of two non-zero values can lead to the presence or the absence of a difference.

3.1 Possible Transitions

Since the S-box is bijective and the `ShiftRows` operation only permutes cells, both those operations do not affect truncated differences. But for the `AddRoundTweakey` and `MixColumns` transformations we need to take care of the

XOR operation. More precisely, given two truncated differences a and b we know that the possible values of $(a, b, a \oplus b)$ are:

$$(0,0,0), (0,1,1), (1,0,1), (1,1,0), (1,1,1)$$

However we have to pay attention to uninstantiable solutions. For instance, given three truncated differences a, b and c, $(1,1,1,0,0,1)$ is a possible value for $(a, b, c, a \oplus b, a \oplus c, b \oplus c)$ but it is impossible to instantiate it because if $a = b$ and $a = c$ then $b = c$.

Hence we rewrite the equation $y = \texttt{MixColumns} \circ \texttt{AddRoundTweakey}(x, k)$ to avoid such patterns:

- $y[1] = x[0] \oplus k[0]$,
- $y[3] = y[1] \oplus x[2]$,
- $y[0] = y[3] \oplus x[3]$,
- $y[2] = x[1] \oplus k[1] \oplus x[3]$

We experimentally verified that each truncated solution of this system can be instantiated.

Keyschedule. When looking at the key schedule of `SKINNY` at the cell level and for truncated differential characteristics it is mostly a simple cell permutation. In the model **SK**, there are no differences in the round keys. In the **TKx** models, differences in the round keys are possible. If the number of rounds targeted is at most 30, the rule for active cells on the round keys is quite simple: either the cell is inactive for all round keys, either it is active for all round keys but one (**TK2**) or two (**TK3**).

3.2 Ad-hoc Models for Step 1

To the best of our knowledge, the most efficient algorithm to search for truncated differential characteristics on SPN ciphers is the one described in [10] by Fouque *et al.* which was applied on the 3 versions of AES. It is mostly dynamic programming as Round i is independent of the paths of rounds $0, 1, \ldots, i-1$ and at each step we only have to save, for each truncated state, the minimal number of active S-boxes to reach it. Hence, the complexity of this algorithm is exponential in the state size but linear in the number of rounds. The algorithm is specified in Algorithm 1. At the end of the algorithm we obtain an array C such that $C[r][s]$ contains the minimal number of active S-boxes required to reach state s after r rounds. Retrieving the truncated representations is then done quite easily using C, starting from the last state to the first. Let say we want to exhaust all truncated differential characteristics on R rounds with at most b active S-boxes ending with state s. From $C[R-1][s]$, we know whether such characteristic exists or not. If $C[R-1][s] \leq b$ we exhaust all states s' such that the transition $s' \rightarrow s$ through one round is possible and, for each of them, we now need to exhaust all truncated differential characteristics on $R-1$ rounds with at most $b - |s|$ active S-boxes ending with state s'.

Algorithm 1: Search for the best truncated representation (**SK**).

```
foreach state s do
  | M[s] ⟵ list of states s' reachable from s through one round
end
foreach state s do
  | C[0][s] ⟵ number of active cells of s
end
for 1 ≤ r < R do
  | foreach state s do C[r][s] ⟵ ∞
  | foreach state s do
  |   | foreach state s' in M[s] do
  |   |   | c ⟵ C[r − 1][s] + number of active cells of s'
  |   |   | if c < C[r][s'] then C[r][s'] ⟵ c
  |   | end
  | end
end
return C
```

The complexity of the algorithm in the single key model is very low, and we experimentally counted around $(R-1) \times 2^{20}$ simple operations for R rounds. A naive solution to search for truncated representations in the **TK1**, **TK2** and **TK3** models would be to apply the previous algorithm for each possible configuration of the key. While for **TK1** this would only increase the overall complexity by a factor 2^{16}, the search would not be practical for both the **TK2** and **TK3** models. Indeed, because of the possible cancellations occurring in the round keys, the number of configurations is very high:

$$\left(\sum_{k=0}^{8}\binom{8}{k}\left(\sum_{i=0}^{tk-1}\binom{\lfloor (R-1)/2\rfloor}{i}\right)^{k}\right)\left(\sum_{k=0}^{8}\binom{8}{k}\left(\sum_{i=0}^{tk-1}\binom{\lceil (R-1)/2\rceil}{i}\right)^{k}\right).$$

For instance, for $R = 30$, there are more than 2^{64} configurations in the **TK2** model.

In the following we present the first practical algorithm which tackles down the problem for the **TK** models without relying on a black box solver as MILP, SAT or CP solvers. Actually this is the only algorithm fast enough to generate all the Step 1 solutions required to perform the Step 2. Indeed, the best differential characteristic is rarely based on the truncated differential characteristics minimizing the number of active S-boxes and thus we need to generate a large number of truncated characteristics to find the one instantiating with the best probability. As we will explain in Sect. 3.4, all other approaches we tried to generate them failed.

The idea of our ad-hoc method is quite similar to the one used in the single key model. Actually, to compute the minimal number of active S-boxes at round $r+1$ we only need to know the minimal number of active S-boxes for each possible state at round r together with the number of cancellations for each

key cell occurred so far. Indeed, we do not need to know at which rounds the cancellations occurred but only how many times they did. A simplified version of this algorithm is described in Algorithm 2. The most important part is related to the variable *cancelled* which count how many times each key cell is cancelled through the encryption. It is a vector of 16 cells, each cell taking values among $\{0, 1, \ldots, x-1, r\}$ for the **TKx** model. The main advantage of our representation is that at each step of the algorithm, $C[r][s]$ contains at most $(x+1)^{16}$ elements for the **TKx** model which is much lower than the number of possible sequences of round keys.

Algorithm 2: Search for the best truncated representation (**TK**).

foreach *state* s, *round key* k **do**
  $M[k][s] \longleftarrow$ list of states s' reachable from s and k through one round
end
foreach *state* s **do**
  $C[0][s] \longleftarrow \{(\text{number of active cells of } s, 0)\}$
end
for $1 \leq r < R$ **do**
  foreach *state* s **do** $C[r][s] \longleftarrow \emptyset$
  foreach *state* s **do**
    foreach (*cost, cancelled*) $\in C[r-1][s]$ **do**
      foreach *round key* k *compatible with cancelled* **do**
        foreach *state* s' *in* $M[k][s]$ **do**
          $c \longleftarrow$ cost + number of active cells of s'
          $C[r][s'] \longleftarrow C[r][s'] \cup \{(c, \text{update}(cancelled, k))\}$
        end
      end
    end
  end
  foreach *state* s **do** keepOptimals($C[r][s]$)
end
return C

Finally we introduce a new improvement which greatly speeds up the search procedure. It is based on the so-called *early abort technique* principle and the idea is to handle the key cell by cell. Indeed, we expect that the best truncated differential characteristics do not involve many active cells in the round key and so we want to quickly cut those branches during the search. To do so we first pick a key cell and guess whether it is active or not. At this step we have not decided yet if any cancellations occur nor their positions but only if it is always 0 or at least once 1. Then we apply the algorithm partially and guess another key cell if and only if it seems possible to find a truncated differential characteristic with a small enough number of active S-boxes. More precisely, along the search we have the relation $y = x \oplus k$ where k is the round key. We introduce a new 16-bit variable g such that $g_i = 0$ if we made a choice for bit i of k and 1

otherwise. To compute the possible truncated transitions from x to y through k for all the possible key (according to g) we can restrict ourself at looking at the possible truncated transitions from $(x|g)$ to y through $(k|g)$ where $|$ is the bitwise OR. Indeed, we use the fact that in truncated setting $1 \oplus 1$ is 0 or 1 and thus our technique allows to handle all the possible keys by looking only at few transitions.

3.3 Results for Step 1

For Step 1, we run our ad-hoc tool on the four attack scenarios (**SK**, **TK1**, **TK2**, and **TK3**) when varying the number of rounds between 3 and 20. We conducted all our experiments on our server composed of 2× AMD EPYC 7742 64-Core and 1TB of RAM. In particular, we were able to complete the security analysis made in [2,3] and claim that the minimal number of active S-boxes in **TK1** for 28, 29 and 30 rounds are 105, 109 and 113 respectively (as shown in Table 1).

Table 1. Lower bounds on the number of active S-boxes in SKINNY.

# Rounds	28	29	30
TK1	105	109	113

However, the optimal solution of Step 2, in terms of differential characteristic probability, could be obtained for a number of active S-boxes which is not the optimal one. Hereafter, we denote Obj_{Step1} the number of active S-boxes we consider when solving the problems. For example, assume that, when processing Step 2, one obtains a differential characteristic with the best probability equal to $2^{-3\times 6} = 2^{-18}$ with $Obj_{Step1} = 6$ and whereas the optimal differential probability of the S-box is 2^{-2}. It means that one has to test all solutions outputted by Step 1 until $Obj_{Step1} < 18/2 = 9$ to be sure that none has a better differential characteristic probability. This is exactly what happened for the case of SKINNY-128 in the **TK** models. We only want to stress here that computing the optimal bounds is often not enough and we need to go further. However, increasing the value of Obj_{Step1} induces an increase of the possible number of Step 1 solutions as illustrated in the third column of Table 4. As one can see, this number of solutions tends to grow exponentially when we increase v. For example, for SKINNY-128 with 14 rounds in the **TK1** model, for the optimal value $v^* = 45$, Step 1 outputs only 3 solutions; whereas we have 897 solutions for $v = v^* + 5 = 50$; 137 019 solutions for $v = v^* + 10 = 55$ and finally 7 241 601 solutions for $v = 59$. So, the time required to output all those Step 1 solutions and the time required for the Step 2 computations on 1 solution outputted by the Step 1 become the bottleneck of the overall process.

3.4 Other Approaches

We tried different approaches to solve the Step 1 problem, including MILP, SAT and CP models.

Our SAT model is encoded through the high level modeling language MiniZinc while our CP model is based on the Choco-solver. Unfortunately, the results of both the SAT and the CP models are really bad: for example, for all instances greater than 16 rounds we were unable to obtain the solutions in reasonable time. This is mainly due to the need to enumerate solutions for SAT, which implies to prohibit all solutions previously found. For CP, on the other hand, this has to do with the nature of the Boolean variables themselves where the Choco-solver can not efficiently propagate lower bounds and upper bounds on Boolean variables.

Our MILP model was much better than our SAT and CP ones. We started from the original model presented in [3] but made several optimizations. First, we added constraints in the **SK** model to obtain all solutions up to column shifts in order to remove symmetries. Moreover, as the original model only describes the way to find the minimal number of active S-boxes, we added a constraint in each model to set a lower bound on the number of active S-boxes and thus, be able to enumerate all the Step 1 solutions given a particular lower bound for the number of active S-boxes. Then, in the original MILP model all xor operations were modeled using dummy variables which is known to be inefficient. Thus we replaced the corresponding inequalities, using that $x \oplus y \oplus z = 0$ can be described with the three inequalities:

$$\{x + y \geq z\},\ \{x + z \geq y\},\ \{y + z \geq x\}.$$

Finally, regarding the resolutions of the MILP models, the parallelization were left to the Gurobi solver.[1]

We compared the MILP model to our ad-hoc tool and we found that our MILP model is much slower in most cases and actually too slow to output all the Step 1 solutions needed to perform Step 2. Running times are given in Table 2.

Table 2. Comparison of the running times required to generate all Step 1 solutions between our MILP and ad-hoc approaches.

Rounds	Model	Obj_{Step1}	MILP	Ad-hoc
14	**TK1**	$45 \rightarrow 59$	>6 h	5 m
19	**TK2**	$52 \rightarrow 63$	>6 h	19 m
20	**SK**	96	342 m	16 s
20	**TK1**	70	38 m	28 s
20	**TK2**	57	745 s	193 s
20	**TK3**	45	998 s	326 s

[1] see: https://www.gurobi.com/documentation/9.0/refman/threads.html .

Note that while our ad-hoc tool gave very good running times, it may require a lot of memory to store the array C. For instance, for 30 rounds in **TK3** mode, our tool required up to 500 GB of RAM to finish the search. It is also important to note that it did not take fully advantage of the 128 cores of our server, and most often used less than 40 cores.

4 Modeling Step 2 with CP

The aim of Step 2 is to try to instantiate the abstracted solutions provided by Step 1 while maximizing the probability of the differential characteristic. Thus, Step 2 takes as input a solution of Step 1 with the objective function of maximizing the probability of the differential characteristic. However, some solutions of Step 1 could not be instantiated in Step 2 as refining the abstraction level of Step 2 will induce *non-consistent* solutions. In the literature, this step has been modeled using ad-hoc methods [6], MILP [1], SAT [28] or CP [12]. As MILP [1] and SAT [28] seem to hardly scale due to prohibitive computational times (linked with the size of the 8-bit S-boxes that must be represented in the form of linear inequalities or of clauses), we focus here on a dedicated CP method implemented using the Choco solver [22]. We also provide, in the second part of this section, the results we obtain when instantiating the differential characteristics in the 4 attack scenarios.

4.1 Constraint Programming

Although less usual than MILP to tackle cryptanalytic problems, CP has already been used in e.g. [9,13]. We recall some basic principles of CP and we refer the reader to [23] for more details.

CP is used to solve Constraint Satisfaction Problems (CSPs). A CSP is defined by a triple (X, D, C) such that $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables, D is a function that maps every variable $x_i \in X$ to its domain $D(x_i)$ and $C = \{c_1, c_2, \ldots, c_m\}$ is a set of constraints. $D(x_i)$ is a finite ordered set of integer values to which the variable x_i can be assigned to, whereas c_j defines a relation between some variables $vars(c_j) \subseteq X$. This relation restricts the set of values that may be assigned simultaneously to $vars(c_j)$. Each constraint is equipped with a filtering algorithm which removes from the domains of $vars(c_j)$, the values that cannot satisfy c_j.

In CP, constraints are classified in two categories. *Extensional constraints*, also called *table constraints*, explicitly define the allowed (or forbidden) tuples of the relation. *Intentional constraints* define the relation using mathematical operators. For instance, in a CSP with $X = \{x_1, x_2, x_3\}$ such that $D(x_1) = D(x_2) = D(x_3) = \{0, 1\}$, a constraint ensuring that the sum of the variables in X is different from 1 can be either expressed in extension (1) or in intention (2):

1. $\textsc{Table}(\langle x_1, x_2, x_3 \rangle, \langle (0,0,0), (0,1,1), (1,0,1), (1,1,0), (1,1,1) \rangle)$
2. $x_1 + x_2 + x_3 \neq 1$

Actually, any intentional constraint can be encoded with an extensional one provided enough memory space, and conversely [8]. However, they may offer different performances.

The purpose of a CSP is to find a *solution*, i.e. an assignment of all variables to a value from their respective domains such that all the constraints are simultaneously satisfied. When looking for a solution, a two-phase mechanism is operated: the *search space exploration* and the *constraint propagation*. The exploration of the search space is processed using a *depth-first search*. At each step, a decision is taken, i.e. a non-assigned variable is selected and its domain is reduced to a singleton. This modification requires to check the satisfiability of all the constraints. This is achieved thanks to constraint propagation which applies each constraint filtering algorithm. Any application may trigger modifications in turn; the propagation ends when either no modification occurs and all constraints are satisfied or a failure is thrown, *i.e.*, at least one constraint cannot be satisfied. In the former case, if all variables are assigned, a solution has been found. Otherwise a new decision is taken and the search is pursued. In the latter case, a backtrack to the first refutable decision is made and the search is resumed.

Turning a CSP into a Constrained Optimisation Problem (COP) is done by adding an objective function. Such a function is defined over variables of X, the purpose is then to find the solution that optimizes the objective function. Finding the optimal solution is done by repeatedly applying the two-phase mechanism above, and by adding a *cut* on the objective function that prevents from finding a same cost solution in the future.

4.2 Modeling Step 2 with CP

Given a Boolean solution for Step 1, Step 2 aims at searching for the byte-consistent solution with the highest (related-tweakey) differential characteristic probability (or proving that there is no byte-consistent solution). In this section, Model 1 describes the CP model we used for `SKINNY`-128 (**SK**). Actually, the ones used to model the other variants, as well as `SKINNY`-64 are rather similar.

For each Boolean variable $\Delta X_{r,i,j}$ of Step 1, we define an integer variable $\delta X_{r,i,j}$. The domain of this integer variable depends on the value of the Boolean variable in the Step 1 solution: If $\Delta X_{r,i,j} = 0$, then the domain is $D(\delta X_{r,i,j}) = \{0\}$ (*i.e.*, $\delta X_{r,i,j}$ is also assigned to 0); otherwise, the domain is $D(\delta X_{r,i,j}) = [1, 255]$. For each byte that passes through an S-box, we define an integer variable $\delta SB_{r,i,j}$ which corresponds to the difference after the S-box. Its domain is $\{0\}$ if $\Delta X_{r,i,j}$ is assigned to 0 in the Step 1 solution; otherwise, it is $D(\delta SB_{r,i,j}) = [1, 255]$. This is expressed in (3) of Model 1.

Finally, as we look for a byte-consistent solution with maximal probability, we also add an integer variable $P_{r,i,j}$ for each byte in an S-box: this variable corresponds to the absolute value of the base 2 logarithm of the probability of the transition through the S-box. Actually, a factor 10 has been applied to avoid considering floats. Thus we define a Table constraint (4) composed of valid triplets of the form $(\delta X_{r,i,j}, \delta SB_{r,i,j}, P_{r,i,j})$. Note that these triplets only

Minimize

$$Obj_{Step2} = \sum_{r=1}^{R}\sum_{i=1}^{4}\sum_{j=1}^{4} P_{r,i,j} \tag{1}$$

subject to

$$20 \times n \leq \sum_{r=1}^{R}\sum_{i=1}^{4}\sum_{j=1}^{4} P_{r,i,j} \leq \min(70 \times n, O^*) \tag{2}$$

$$\forall r \in 1..R, \forall i \in 1..4, \forall j \in 1..4$$
$$\begin{cases} \delta X_{r,i,j} = 0 \wedge \delta SB_{r,i,j} = 0 \wedge P_{r,i,j} = 0 & \text{if } \Delta X_{r,i,j} = 0 \\ \delta X_{r,i,j} \geq 1 \wedge \delta SB_{r,i,j} \geq 1 \wedge P_{r,i,j} \geq 20 & \text{otherwise} \end{cases} \tag{3}$$

$$\forall r \in 1..R, \forall i \in 1..4, \forall j \in 1..4$$
$$\textsc{Table}(\langle \delta X_{r,i,j}, \delta SB_{r,i,j}, P_{r,i,j} \rangle, \langle \text{SBox} \rangle) \quad \text{if } \Delta X_{r,i,j} \neq 0 \tag{4}$$

$$\forall r \in 1..R-1, \forall j \in 1..4 \qquad \delta SB_{r,0,j} = \delta X_{r+1,1,j} \tag{5}$$

$$\forall r \in 1..R-1, \forall j \in 1..4$$
$$\begin{cases} \delta SB_{r,2,(2+j)\%4} = \delta X_{r+1,2,j} & \text{if } \Delta SB_{r,1,(3+j)\%4} = 0 \\ \delta SB_{r,1,(3+j)\%4} = \delta X_{r+1,2,j} & \text{if } \Delta SB_{r,2,(2+j)\%4} = 0 \\ \delta SB_{r,1,(3+j)\%4} = \delta SB_{r,2,(2+j)\%4} & \text{if } \Delta X_{r+1,2,j} = 0 \\ \textsc{Table}(\langle \delta SB_{r,1,(3+j)\%4}, \delta SB_{r,2,(2+j)\%4}, \delta X_{r+1,2,j} \rangle, \langle \text{XOR} \rangle) & \text{otherwise} \end{cases} \tag{6}$$

$$\forall r \in 1..R-1, \forall j \in 1..4$$
$$\begin{cases} \delta SB_{r,2,(2+j)\%4} = \delta X_{r+1,3,j} & \text{if } \Delta SB_{r,0,j} = 0 \\ \delta SB_{r,0,j} = \delta X_{r+1,3,j} & \text{if } \Delta SB_{r,2,(2+j)\%4} = 0 \\ \delta SB_{r,0,j} = \delta SB_{r,2,(2+j)\%4} & \text{if } \Delta X_{r+1,3,j} = 0 \\ \textsc{Table}(\langle \delta SB_{r,0,j}, \delta SB_{r,2,(2+j)\%4}, \delta X_{r+1,3,j} \rangle, \langle \text{XOR} \rangle) & \text{otherwise} \end{cases} \tag{7}$$

$$\forall r \in 1..R-1, \forall j \in 1..4$$
$$\begin{cases} \delta X_{r+1,0,j} = \delta X_{r+1,3,j} & \text{if } \Delta SB_{r,3,(1+j)\%4} = 0 \\ \delta SB_{r,3,(1+j)\%4} = \delta X_{r+1,3,j} & \text{if } \Delta X_{r+1,0,j} = 0 \\ \delta SB_{r,3,(1+j)\%4} = \delta X_{r+1,0,j} & \text{if } \Delta X_{r+1,3,j} = 0 \\ \textsc{Table}(\langle \delta SB_{r,3,(1+j)\%4}, \delta X_{r+1,0,j}, \delta X_{r+1,3,j} \rangle, \langle \text{XOR} \rangle) & \text{otherwise} \end{cases} \tag{8}$$

where $\forall r \in R..n$, $\forall i \in 1..4$, $\forall j \in 1..4$,

$$\delta X_{r,i,j} \in 0..255,\ \delta SB_{r,i,j} \in 0..255,\ P_{r,i,j} \in \{0, 20, .., 70\},$$

and $\langle \text{XOR} \rangle$ encodes $\oplus$ relation and $\langle \text{SBox} \rangle$ the S-box constraint.

Model 1: Formulation of **SK** Step2.

contain non-zero values and that $P_{r,i,j}$ takes only 2 different values for the 4-bit S-box (SKINNY-64) and 7 different values for the 8-bit S-box (SKINNY-128). There are roughly 2^{14} triplet elements in the Table constraint for the SKINNY-128 case. As the S-box layer is the only non-linear layer, the other operations could be directly implemented in a deterministic way at the cell level. The associated constraints thus follow the SKINNY-128 linear operations. When possible, i.e. when one element is known to be zero, we replace XOR constraints (encoded using TABLEconstraints) by a simple equality constraint. This corresponds to TABLE constraints (5), (6), (7) and (8) in Model 1.

The overall goal is finally to find a byte-consistent solution which maximizes differential characteristic probability. Thus, we define an integer variable Obj_{Step2} to minimize the sum of all $P_{r,i,j}$ variables (1). This value mainly depends on the number of S-boxes outputted by Step1 Obj_{Step1} and can be bounded to $[\![20 \cdot Obj_{Step1}, 70 \cdot Obj_{Step1}]\!]$ (2).

The differences for the models **TK1**, **TK2** and **TK3** are the modeling of the XORs induced by the lanes of the tweakey through XOR table constraints. Each XOR constraint depicted in Model 1 provides high quality filtering but requires 65536 tuples to be stored which results in prohibitive memory usage. This may limit the number of threads that can be used for the resolution, which was the case for **TK2** and **TK3**. To get around this issue, we encoded the XOR constraint in intention (by defining filtering rules), providing a more memory efficient algorithm, at the expense of filtering strength. This last choice was applied for **TK2** and **TK3** (SKINNY-128 only). We also rely on TABLEconstraints to model the LFSRs applied on TK2 and TK3.

Concerning the search space strategy, for the **TK2** and the **TK3** attack settings, the Step 1 only outputs the truncated value of the sum of the TKi. Thus, the search space strategy first looks at the cancellation places of the sum of the TKi and then instantiates the TKi values according to those positions. For the **TK1** setting, we simply apply the default Choco-solver strategy.

Concerning the parallelization, we affect one solution outputted by Step 1 per thread and we share between the threads the value of Obj_{Step2}.

4.3 Step 2 Performance Results

We run our Step 2 model on the two versions of SKINNY (SKINNY-64 and SKINNY-128) using our CP models written in Choco-solver. We conduct all our experiments on our server composed of 2× AMD EPYC 7742 64-Core and 1TB of RAM. All the reported times are `real` system times.

Up to our knowledge, we only found [1] that gives time results concerning finding the best **SK** differential characteristic probability on SKINNY-128 using a MILP tool based on Gurobi.

More precisely, the authors say: "In our experiments, we used Gurobi Optimizer with Xeon Processor E5-2699 (18 cores) in 128 GB RAM." and, for 13 rounds, "in our environment, the test of 6 classes [Step 1 solutions with 58 active S-boxes without symmetries] finished in 16 days. Finally, it is proven that

the tight bound on the probability of differential characteristic for 13 rounds is 2^{-123}" in the **SK** model.

Regarding the **TK** models, the best known results were obtained by Liu *et al.* also using MILP models [19]. They could only find the best differential characteristics up to 7, 9 and 13 rounds for **TK1**, **TK2** and **TK3** respectively.

Results for SKINNY-64. We sum up in Table 3 all the results we obtain for SKINNY-64 in the four different attack models (**SK**,**TK1**,**TK2** and **TK3**). The overall time, in this case, is not a bottleneck. We only give results concerning number of rounds that are at the limit (just under and just upper) when regarding the number of active S-boxes which is equal to 32 in the case of SKINNY-64 as the state size is 64 bits and as the best differential probability of the S-box is equal to 2^{-2}. Thus, the best overall differential characteristic probability must be under 2^{-64}.

Note that sometimes, we need to browse several Obj_{Step1} bounds to find the optimal differential characteristic probability when the number of rounds is fixed. Indeed, we need to proactively adapt the probability bound we found. For example, in the case of **TK2** SKINNY-64 with 13 rounds, the optimal Obj_{Step1} is equal to 25 and when providing the Step 2 process with this Obj_{Step1} bound, we find a best differential characteristic probability equal to 2^{-55}. Thus, we need to enumerate all the Step 1 solutions with $Obj_{Step1} = 26$ and $Obj_{Step1} = 27$ to be sure that the previous probability is really the best one. Then, before running again Step 2 on those new results we adapt the best probability to the new bound equal to 2^{-55} instead of the old bound equal to 2^{-64}.

We also provide in Appendix A the details of the best found differential characteristics.

Table 3. Overall results concerning SKINNY-64 in the four attack models. Step 2 time corresponds to the Step 2 time taken over all Step 1 solutions when Obj_{step1} takes the values precise in the first column. Best Pr corresponds to the best found probability of a differential characteristic.

	Nb Rounds	Obj_{Step1}	Nb sol. Step 1	Step 2 time	Best Pr
SK	7	26	2	1 s	2^{-52}
SK	8	36	17	1 s	$<2^{-64}$
TK1	10	23	1	1 s	2^{-46}
TK1	11	32	2	1 s	$=2^{-64}$
TK2	13	25 → 27	10	1 s	2^{-55}
TK2	14	31	1	1 s	$<2^{-64}$
TK3	15	24 → 26	46	2 s	2^{-54}
TK3	16	27 → 31	87	4 s	$=2^{-64}$
TK3	17	31	2	1 s	$<2^{-64}$

Results for SKINNY-128. In the same way, we provide in Table 4 the best differential characteristic probability with the total time required for this search for the 4 different attack models. As one can see, we also verify all the possible values for Obj_{Step1} for a given number of rounds, depending on the probability value previously found. Thus, this time, the number of solutions outputted by Step 1 could be huge when we move away from the optimal Step 1 value v^*. However, as the time spent to solve one solution is reasonable (at least when considering **SK** and **TK1**), our model scales reasonably well: the worst case requires 25 days of `real` time on our server on 8 threads and 31 GB of RAM[2].

Table 4. Overall results concerning SKINNY-128 in the four attack models. Step 2 time corresponds to the Step 2 time taken over all solutions of *Step1-enum* when Obj_{step1} takes the values precise in the first column. Best Pr corresponds to the best found probability of a differential characteristic.

	Nb Rounds	Obj_{step1}	Nb sol. Step 1	Step 2 time	Best Pr
SK	9	41 → 43	52	16 s	2^{-86}
SK	10	46 → 48	48	11 s	2^{-96}
SK	11	51 → 52	15	4 s	2^{-104}
SK	12	55 → 56	11	6 s	2^{-112}
SK	13	58 → 61	18	2 m 27 s	2^{-123}
SK	14	61 → 63	6	21 s	$\leq 2^{-128}$
TK1	8	13 → 16	14	4 s	2^{-33}
TK1	9	16 → 20	6	3 s	2^{-41}
TK1	10	23 → 27	6	4 s	2^{-55}
TK1	11	32 → 36	531	37 s	2^{-74}
TK1	12	38 → 46	186 482	213 m	2^{-93}
TK1	13	41 → 53	2 385 482	2 days	$2^{-106.2}$
TK1	14	45 → 59	11 518 612	20 days	2^{-120}
TK1	15	49 → 63	7 542 053	25 days	$\leq 2^{-128}$
TK2	9	9 → 10	7	3 s	2^{-20}
TK2	10	12 → 17	132	11 s	$2^{-34.4}$
TK2	11	16 → 25	4203	6 m	$2^{-51.4}$
TK2	12	21 → 35	1 922 762	512 m	$2^{-70.4}$
TK2	19	52 → 63	530 693	280 m	$\leq 2^{-128}$
TK3	10	6	3	3 s	2^{-12}
TK3	11	10	3	10 s	2^{-21}
TK3	12	13 → 17	373	1 h	$2^{-35.7}$
TK3	13	16 → 25	34 638	85 h	$2^{-51.8}$
TK3	23	55 → 63	47 068	11 h	$\leq 2^{-128}$

[2] It seems that the use of the 128 threads was prohibited by the memory usage of XOR tables (i.e. XOR in extension).

Table 5. Overall results concerning SKINNY-128 with exactly one active cell in the tweakey.

	Nb Rounds	Obj_{step1}	Best Pr
TK2	13	$25 \rightarrow 44$	$2^{-86.2}$
TK2	14	$31 \rightarrow 54$	$\geq 2^{-105.8}$
TK2	15	$35 \rightarrow 56$	$\geq 2^{-113.8}$
TK2	16	$40 \rightarrow 63$	$\geq 2^{-127.6}$
TK3	14	$19 \rightarrow 33$	2^{-67}
TK3	15	$24 \rightarrow 40$	2^{-81}
TK3	16	$27 \rightarrow 48$	2^{-98}
TK3	17	$31 \rightarrow 54$	2^{-110}
TK3	19	$43 \rightarrow 63$	$\leq 2^{-128}$
TK3	20	$45 \rightarrow 63$	$\leq 2^{-128}$
TK3	21	$48 \rightarrow 63$	$\leq 2^{-128}$
TK3	22	$51 \rightarrow 63$	$\leq 2^{-128}$

Our **TK2** and **TK3** models are based on XOR constraints encoded in intention (and not using tables) and these experiences have been launched using the 128 threads of our server.

Concerning **TK2** and **TK3**, we were not able to perform all the computations due to the huge number of Step 1 solutions. Hence we decided to handle only the Step 1 solutions with exactly one active byte in the round keys in order to limit the number of truncated characteristics to instantiate. Those results are given in Table 5. We provide in Appendix B the best **TK2** differential characteristic we found for 16 rounds, and the best **TK3** differential characteristic we found for 17 rounds. Note that we do not know if these differential characteristics are optimal in terms of probability as we were not able to test all the solutions Step 1.

Lessons Learnt. The overall gap is not to find the optimal value of $Obj_{Step1} = v^*$ for a given number of rounds and to enumerate the corresponding overall solutions if the Step 1 model is sufficiently tight. The real gap is if the value obtained for Obj_{Step2} (here equal to $2 \times v^*$ as the best differential probability for the S-box is equal to 2^{-2}) is far from the optimal bound then we have to increase Obj_{Step1} up to the bound $\lfloor Obj_{Step2}/2 \rfloor$. Further we are from v^* in the Step 1 resolution, more numerous are the Step 1 solutions (in fact this number grows exponentially as could be seen in Table 4). Thus, the time for the Step 2 resolution becomes the bottleneck.

5 Conclusion

In this paper, we improve the security bounds regarding differential characteristics search on the block cipher SKINNY. As usually done, we have divided the

search procedure into two steps: Step 1 which abstracts the difference values into Boolean variables and finds the truncated characteristics with the smallest number of active S-boxes; and Step 2 which inputs the results of Step 1 to output the best possible probability instantiating the abstract solutions outputted by Step 1. Of course, each solution of Step 1 could not always be instantiated in Step 2.

For Step 1, an ad-hoc method which heavily uses the structure of the problem is proposed. For solving Step 2, we have implemented a Choco-solver model. Regarding Step 2, our Choco-solver model is much faster than any other approaches. It allowed us to find, for the first time, the best (related-tweakey) differential characteristics in the **TK1** model up to 14 rounds for SKINNY-128 and to show there is no differential trail on 15 rounds with a probability better than 2^{-128}. Regarding the **TK2** model, we were able to find the best differential trails up to 16 rounds. For **TK3**, we are able to exhibit a differential characteristic up to 17 rounds. Note that in [19] Liu *et al.* were only able to reach 7 and 9 rounds in the **TK1** and **TK2** model respectively. Our approach is thus an important improvement.

A Best (Related-Tweakey) Differential Characteristics for SKINNY-64

The best **SK** differential characteristics on 7 rounds of SKINNY-64 with probability equal to 2^{-52} is given in Table 6. The best **TK1** differential characteristics on 10 rounds of SKINNY-64 with probability equal to 2^{-46} is given in Table 7. The Best **TK2** differential characteristics on 13 rounds of SKINNY-64 with probability equal to 2^{-55} is given in Table 8. Best **TK3** differential characteristics on 15 rounds of SKINNY-64 with probability equal to 2^{-54} is given in Table 9.

Table 6. The Best **SK** differential characteristics on 7 rounds of SKINNY-64 with probability equal to 2^{-52}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB)	δSBX_i (after SB)	Pr(States)
$i = 1$	0040 4444 4440 4400	0020 2222 2220 2200	$2^{-2 \cdot 10}$
2	0000 0020 0200 2002	0000 0010 0100 1001	$2^{-2 \cdot 4}$
3	0010 0000 0000 0001	0080 0000 0000 0008	$2^{-2 \cdot 2}$
4	0000 0080 0000 0080	0000 0040 0000 0040	$2^{-2 \cdot 2}$
5	0400 0000 0004 0000	0200 0000 0002 0000	$2^{-2 \cdot 2}$
6	0000 0200 0200 0000	0000 0100 0100 0000	$2^{-2 \cdot 2}$
7	0001 0000 0011 0001	0008 0000 0088 0008	$2^{-2 \cdot 4}$

Table 7. The Best **TK1** differential characteristics on 10 rounds of SKINNY-64 with probability equal to 2^{-46}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB)	δSBX_i (after SB)	$\delta TK1_i$	Pr(States)
$i = 1$	0000 0002 0020 0200	0000 0001 0010 0100	1000 0000 0B80 0000	$2^{-2 \cdot 3}$
2	1000 1000 0000 0000	B000 8000 0000 0000	B000 8000 1000 0000	$2^{-2 \cdot 2}$
3	0000 0000 0000 0000	0000 0000 0000 0000	0010 0000 B000 8000	–
4	0010 0010 0000 0010	00B0 00A0 0000 00B0	00B0 0080 0010 0000	$2^{-2 \cdot 3}$
5	0B00 0000 0002 0000	0100 0000 0001 0000	0000 1000 00B0 0080	$2^{-2 \cdot 2}$
6	0000 0100 0000 0000	0000 0800 0000 0000	0000 B800 0000 1000	$2^{-2 \cdot 1}$
7	0000 0000 0B00 0000	0000 0000 0100 0000	0000 0010 0000 B800	$2^{-2 \cdot 1}$
8	0001 0000 0000 0001	0008 0000 0000 0008	0008 00B0 0000 0010	$2^{-2 \cdot 2}$
9	0080 0000 000B 0000	0040 0000 0001 0000	0000 0100 0008 00B0	$2^{-2 \cdot 2}$
10	0140 0040 0110 0140	0820 0020 0880 0820	0000 0B08 0000 0100	$2^{-2 \cdot 7}$

Table 8. The Best **TK2** differential characteristics on 13 rounds of SKINNY-64 with probability equal to 2^{-55}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB)	δSBX_i (after SB)	$\delta TK1_i$	$\delta TK2_i$	Pr(States)
$i = 1$	0000 8200 0080 0000	0000 4100 0040 0000	0000 0008 0502 0000	0000 000C 060C 0000	$2^{-2 \cdot 3}$
2	4000 0000 0410 4000	2000 0000 02A0 2000	5000 0002 0000 0008	D000 0008 0000 000C	$2^{-2 \cdot 4}$
3	0000 A000 0002 0002	0000 6000 0006 0003	0800 0000 5000 0002	0800 0000 D000 0008	$2^{-2 \cdot 3}$
4	0630 0000 0000 0600	03F0 0000 0000 0100	0250 0000 0800 0000	01A0 0000 0800 0000	$2^{-3 \cdot 3}$
5	1000 0000 0000 0000	9000 0000 0000 0000	8000 0000 0250 0000	1000 0000 01A0 0000	2^{-2}
6	0000 0000 0000 0000	0000 0000 0000 0000	2000 5000 8000 0000	2000 5000 1000 0000	–
7	0000 0000 0000 0000	0000 0000 0000 0000	0080 0000 2000 5000	0020 0000 2000 5000	–
8	00A0 00A0 0000 00A0	0060 0050 0000 0050	0020 0050 0080 0000	0040 00B0 0020 0000	$2^{-2 \cdot 3}$
9	0500 0000 000B 0000	0C00 0000 000C 0000	0000 8000 0020 0050	0000 4000 0040 00B0	$2^{-3 \cdot 2}$
10	0000 0C00 0000 0000	0000 0200 0000 0000	0000 2500 0000 8000	0000 9700 0000 4000	2^{-2}
11	0000 0000 0B00 0000	0000 0000 0100 0000	0000 0080 0000 2500	0000 0090 0000 9700	2^{-2}
12	0001 0000 0000 0001	000A 0000 0000 0008	0005 0020 0000 0080	000F 0030 0000 0090	$2^{-2 \cdot 2}$
13	0080 0000 0001 0000	0040 0000 0008 0000	0000 0800 0005 0020	0000 0300 000F 0030	$2^{-2 \cdot 2}$

Table 9. The Best **TK3** differential characteristics on 15 rounds of SKINNY-64 with probability equal to 2^{-54}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB)	δSBX_i (after SB)	$\delta TK1_i$	$\delta TK2_i$	$\delta TK3_i$	Pr(States)
$i = 1$	0000 0001 4000 0004	0000 0008 2000 0002	0000 080D 0000 0800	0000 0408 0000 0500	0000 0E0D 0000 0C00	$2^{-2.3}$
2	0000 0000 0000 0020	0000 0000 0000 0010	0008 0000 0000 080D	000B 0000 0000 0408	000E 0000 0000 0E0D	2^{-2}
3	010D 000D 0000 000D	0A0E 0002 0000 0002	0D08 0000 0008 0000	0109 0000 000B 0000	060F 0000 000E 0000	$2^{-2.3}2^{-3}$
4	0020 0000 2000 0000	0030 0000 3000 0000	0000 0008 0D08 0000	0000 0007 0109 0000	0000 000F 060F 0000	$2^{-2.2}$
5	0000 0030 0030 0000	0000 00C0 00C0 0000	D000 0008 0000 0008	2000 0003 0000 0007	3000 0007 0000 000F	$2^{-3.2}$
6	0000 C000 000C 0000	0000 2000 0002 0000	0800 0000 D000 0008	0F00 0000 2000 0003	0700 0000 3000 0007	$2^{-2.2}$
7	0200 0000 0000 0200	0500 0000 0000 0300	08D0 0000 0800 0000	0640 0000 0F00 0000	0B90 0000 0700 0000	$2^{-2.2}$
8	3000 0000 0000 0000	D000 0000 0000 0000	8000 0000 08D0 0000	E000 0000 0640 0000	B000 0000 0B90 0000	2^{-3}
9	0000 0000 0000 0000	0000 0000 0000 0000	8000 D000 8000 0000	D000 9000 E000 0000	5000 4000 B000 0000	–
10	0000 0000 0000 0000	0000 0000 0000 0000	0080 0000 8000 D000	00C0 0000 D000 9000	0050 0000 5000 4000	–
11	0010 0010 0000 0010	0080 0090 0000 00A0	0080 00D0 0080 0000	00A0 0030 00C0 0000	00A0 0020 0050 0000	$2^{-2.3}$
12	0A00 0000 0005 0000	0A00 0000 000A 0000	0000 8000 0080 00D0	0000 8000 00A0 0030	0000 A000 00A0 0020	$2^{-2}2^{-3}$
13	0000 0A00 0000 0000	0000 0A00 0000 0000	0000 8D00 0000 8000	0000 5600 0000 8000	0000 D100 0000 A000	2^{-3}
14	0000 0000 0000 0000	0000 0000 0000 0000	0000 0080 0000 8D00	0000 0010 0000 5600	0000 00D0 0000 D100	–
15	0000 0000 0004 0000	0000 0000 0002 0000	000D 0080 0000 0080	000D 00B0 0000 0010	0008 0060 0000 00D0	2^{-2}

B Best (Related-Tweakey) Differential Characteristics for SKINNY-128

Concerning the best **SK** differential characteristics on 13 rounds of SKINNY-128, We obtain the same best **SK** differential characteristics on 13 rounds of SKINNY-128 with probability equal to 2^{-123} given in Table 11 of Appendix D of [1]. The best **TK1** differential characteristics on 14 rounds of SKINNY-128 with probability equal to 2^{-120} is given in Table 10. The best **TK2** differential characteristics on 16 rounds of SKINNY-128 with probability equal to $2^{-127.6}$ we found is given in Table 11. The best **TK3** differential characteristics on 17 rounds of SKINNY-128 with probability equal to 2^{-110} we found is given in Table 12.

Table 10. The Best **TK1** differential characteristics on 14 rounds of SKINNY-128 with probability equal to 2^{-120}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB)	δSBX_i (after SB)	$\delta TK1_i$	Pr(States)
$i=1$	02000002 00000200 00020000 00020040	08000008 00000800 00080000 00080004	00000000 00000000 01000000 00000000	$2^{-2\cdot 6}$
2	00000400 08000008 00000000 08000000	00000100 10000010 00000000 10000000	00000100 00000000 00000000 00000000	$2^{-2\cdot 4}$
3	00000010 00000000 10100000 00000000	00000040 00000000 40400000 00000000	00000000 00000000 00000100 00000000	$2^{-2\cdot 3}$
4	00004000 00000040 00004040 00004000	00000400 00000004 00000404 00000400	00000000 01000000 00000000 00000000	$2^{-2\cdot 5}$
5	04000400 00000400 00050000 04040400	05000500 00000100 00050000 05050500	00000000 00000000 00000000 01000000	$2^{-3\cdot 6}2^{-2}$
6	00050500 05000500 00000004 05000505	00050500 01000100 00000005 05000505	00000000 00000100 00000000 00000000	$2^{-3\cdot 6}2^{-2\cdot 2}$
7	00050005 00050500 00040000 00000500	00050005 00050500 00050000 00000500	00000000 00000000 00000000 00000100	$2^{-3\cdot 6}$
8	00000000 00050005 00000500 00050000	00000000 00010005 00000500 00050000	00000000 00010000 00000000 00000000	$2^{-3\cdot 3}2^{-2}$
9	00000000 00000000 00000000 05000000	00000000 00000000 00000000 05000000	00000000 00000000 00000000 00010000	2^{-3}
10	00000005 00000000 00000000 00000000	00000001 00000000 00000000 00000000	00000001 00000000 00000000 00000000	2^{-2}
11	00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000	00000000 00000000 00000001 00000000	–
12	00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000	00000000 00000001 00000000 00000000	–
13	00000000 00000000 01000000 00000000	00000000 00000000 20000000 00000000	00000000 00000000 00000000 00000001	2^{-2}
14	00002000 00000000 00002000 00002000	00008000 00000000 00008000 00008000	00010000 00000000 00000000 00000000	$2^{-2\cdot 3}$

Table 11. The Best **TK2** differential characteristics we found on 16 rounds of SKINNY-128 with probability equal to $2^{-127.6}$. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB) δSBX_i (after SB)	$\delta TK1_i$ $\delta TK2_i$	Pr(States)
$i=1$	00000000 00404010 40400000 40000000 00000000 00040440 04040000 04000000	00000000 00000000 00000000 00007700 00000000 00000000 00000000 00003900	$2^{-2\cdot 6}$
2	00000400 00000000 40000000 00000404 00000500 00000000 04000000 00000101	00000000 00770000 00000000 00000000 00000000 00730000 00000000 00000000	$2^{-2\cdot 3}2^{-3}$
3	00010000 00000500 00000000 00000100 00200000 00000500 00000000 00002000	00000000 00000000 00000000 00770000 00000000 00000000 00000000 00730000	$2^{-2\cdot 2}2^{-3}$
4	00000000 00200000 00000005 00200000 00000000 00800000 00000005 00800000	00000077 00000000 00000000 00000000 000000E7 00000000 00000000 00000000	$2^{-2\cdot 2}2^{-3}$
5	80050090 00000090 00058000 00050090 03010002 00000002 00010200 00010003	00000000 00000000 00000077 00000000 00000000 00000000 000000E7 00000000	$2^{-2\cdot 8}$
6	00010303 03010002 00000001 01010003 00202020 20200009 00000020 20200020	00000000 00000077 00000000 00000000 00000000 000000CE 00000000 00000000	$2^{-2\cdot 6}2^{-3\cdot 4}$
7	20000000 00202020 B0002000 00002020 80000000 00808080 80008000 00009380	00000000 00000000 00000000 00000077 00000000 00000000 00000000 000000CE	$2^{-2\cdot 6}2^{-2\cdot 4}2^{-3}$
8	00930000 80000000 00000080 00008000 00EA0000 03000000 00000003 00000300	00770000 00000000 00000000 00000000 009D0000 00000000 00000000 00000000	$2^{-2\cdot 3}2^{-6}$
9	00000000 00000000 00000000 00030000 00000000 00000000 00000000 00BC0000	00000000 00000000 00770000 00000000 00000000 00000000 009D0000 00000000	2^{-5}
10	BC000000 00000000 00000000 00000000 4C000000 00000000 00000000 00000000	77000000 00000000 00000000 00000000 3B000000 00000000 00000000 00000000	2^{-6}
11	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 77000000 00000000 00000000 00000000 3B000000 00000000	–
12	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00007700 00000000 00000000 00000000 00007700 00000000 00000000 00000000	–
13	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 00007700 00000000 00000000 00000000 00007700 00000000	–
14	0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 77000000 00000000 00000000 00000000 EF000000 00000000 00000000	–
15	00000000 00000000 00980000 00000000 00000000 00000000 00420000 00000000	00000000 00000000 00000000 77000000 00000000 00000000 00000000 EF000000	2^{-5}
16	00000042 00000000 00000042 00000042 00000008 00000000 00000008 00000008	–	$2^{-2\cdot 4\cdot 3}$

Table 12. The Best **TK3** differential characteristics we found on 17 rounds of SKINNY-128 with probability equal to 2^{-110}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB) δSBX_i (after SB)	$\delta TK1_i$ $\delta TK2_i$ $\delta TK3_i$	Pr(States)
$i = 1$	00000200 00320000 08000000 00000808 00000800 00920000 18000000 00001010	00000000 00BA0000 00000000 00000000 00000000 00430000 00000000 00000000 00000000 00730000 00000000 00000000	$2^{-2\cdot3}2^{-3\cdot2}$
2	00100000 00000800 00000000 00001000 00400000 00001000 00000000 00004000	00000000 00000000 00000000 00BA0000 00000000 00000000 00000000 00430000 00000000 00000000 00000000 00730000	$2^{-2\cdot3}$
3	00000000 00400000 00000010 00400000 00000000 00040000 00000040 00040000	000000BA 00000000 00000000 00000000 00000086 00000000 00000000 00000000 00000039 00000000 00000000 00000000	$2^{-2\cdot3}$
4	04400005 00000005 00400400 00400005 05040001 00000001 00040100 00040005	00000000 00000000 000000BA 00000000 00000000 00000000 00000086 00000000 00000000 00000000 00000039 00000000	$2^{-2\cdot6}2^{-3\cdot2}$
5	00040505 05040001 00000004 04040005 00010101 01010028 00000001 01010001	00000000 000000BA 00000000 00000000 00000000 0000000D 00000000 00000000 00000000 0000009C 00000000 00000000	$2^{-2\cdot9}2^{-3}$
6	01000000 00010101 03000100 00000101 20000000 00202020 20002000 0000B320	00000000 00000000 00000000 000000BA 00000000 00000000 00000000 0000000D 00000000 00000000 00000000 0000009C	$2^{-2\cdot6}2^{-3}2^{-4}$
7	00B30000 20000000 00000020 00002000 00EE0000 80000000 00000080 00008000	00BA0000 00000000 00000000 00000000 001A0000 00000000 00000000 00000000 004E0000 00000000 00000000 00000000	$2^{-2\cdot3}2^{-7}$
8	00000000 00000000 00000000 00800000 00000000 00000000 00000000 00030000	00000000 00000000 00BA0000 00000000 00000000 00000000 001A0000 00000000 00000000 00000000 004E0000 00000000	2^{-2}
9	03000000 00000000 00000000 00000000 29000000 00000000 00000000 00000000	BA000000 00000000 00000000 00000000 34000000 00000000 00000000 00000000 A7000000 00000000 00000000 00000000	2^{-4}
10	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 BA000000 00000000 00000000 00000000 34000000 00000000 00000000 00000000 A7000000 00000000	–
11	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	0000BA00 00000000 00000000 00000000 00006900 00000000 00000000 00000000 0000D300 00000000 00000000 00000000	–
12	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 0000BA00 00000000 00000000 00000000 00006900 00000000 00000000 00000000 0000D300 00000000	–
13	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 BA000000 00000000 00000000 00000000 D3000000 00000000 00000000 00000000 69000000 00000000 00000000	–
14	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 00000000 BA000000 00000000 00000000 00000000 D3000000 00000000 00000000 00000000 69000000	–
15	0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 0000BA00 00000000 00000000 00000000 0000A700 00000000 00000000 00000000 00003400 00000000 00000000	–
16	00000000 00000000 00000029 00000000 00000000 00000000 00000030 00000000	00000000 00000000 00000000 0000BA00 00000000 00000000 00000000 0000A700 00000000 00000000 00000000 00003400	2^{-3}
17	00300000 00000000 00300000 00300000 00400000 00000000 00400000 00400000	–	$2^{-2\cdot3}$

References

1. Abdelkhalek, A., Sasaki, Y., Todo, Y., Tolba, M., Youssef, A.M.: MILP modeling for (large) s-boxes to optimize probability of differential characteristics. IACR Trans. Symmetric Cryptol. **2017**(4), 99–129 (2017)
2. Alfarano, G.N., Beierle, C., Isobe, T., Kölbl, S., Leander, G.: ShiftRows alternatives for AES-like ciphers and optimal cell permutations for Midori and SKINNY. IACR Trans. Symmetric Cryptol. **2018**(2), 20–47 (2018). https://doi.org/10.13154/tosc.v2018.i2.20-47
3. Beierle, C., et al.: The `SKINNY` family of block ciphers and its low-latency variant `MANTIS`. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_5
4. Biham, E.: New types of cryptanalytic attacks using related keys. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 398–409. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_34
5. Biham, E., Shamir, A.: Differential cryptanalysis of feal and n-hash. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 1–16. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_1
6. Biryukov, A., Nikolić, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: application to AES, Camellia, Khazad and others. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 322–344. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_17
7. Delaune, S., Derbez, P., Huynh, P., Minier, M., Mollimard, V., Prud'Homme, C.: SKINNY with scalpel comparing tools for differential analysis (April 2021). https://hal.archives-ouvertes.fr/hal-03040548, working paper or preprint
8. Demeulenaere, J., et al.: Compact-table: efficiently filtering table constraints with reversible sparse bit-sets. In: Rueher, M. (ed.) CP 2016. LNCS, vol. 9892, pp. 207–223. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44953-1_14
9. Eichlseder, M., Nageler, M., Primas, R.: Analyzing the linear keystream biases in AEGIS. IACR Trans. Symmetric Cryptol. **2019**(4), 348–368 (2019)
10. Fouque, P.-A., Jean, J., Peyrin, T.: Structural evaluation of AES, and chosen-key distinguisher of 9-Round AES-128. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 183–203. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_11
11. Gérault, D., Lafourcade, P., Minier, M., Solnon, C.: Revisiting AES related-key differential attacks with constraint programming. Inf. Process. Lett. **139**, 24–29 (2018)
12. Gerault, D., Lafourcade, P., Minier, M., Solnon, C.: Computing AES related-key differential characteristics with constraint programming. Artif. Intell. **278**, 103183 (2020)
13. Gerault, D., Minier, M., Solnon, C.: Constraint programming models for chosen key differential cryptanalysis. In: Rueher, M. (ed.) CP 2016. LNCS, vol. 9892, pp. 584–601. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44953-1_37
14. Jean, J.: TikZ for cryptographers (2016). https://www.iacr.org/authors/tikz/
15. Jean, J., Nikolić, I., Peyrin, T.: Tweaks and keys for block ciphers: the `TWEAKEY` framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_15
16. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60590-8_16

17. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 161–185. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_8
18. Lafitte, F.: Cryptosat: a tool for sat-based cryptanalysis. IET Inf. Secur. **12**(6), 463–474 (2018)
19. Liu, G., Ghosh, M., Song, L.: Security analysis of SKINNY under related-tweakey settings (long paper). IACR Trans. Symmetric Cryptol. **2017**(3), 37–72 (2017). https://doi.org/10.13154/tosc.v2017.i3.37-72
20. Matsui, M.: On correlation between the order of S-boxes and the strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0053451
21. Mouha, N., Preneel, B.: A proof that the ARX cipher salsa20 is secure against differential cryptanalysis. IACR Cryptol. ePrint Arch. **2013**, 328 (2013). http://eprint.iacr.org/2013/328
22. Prud'homme, C., Fages, J.G., Lorca, X.: Choco documentation. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S. (2016). http://www.choco-solver.org
23. Rossi, F., Beek, P.V., Walsh, T.: Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc., New York (2006)
24. Sasaki, Yu., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 185–215. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_7
25. Song, L., Qin, X., Hu, L.: Boomerang connectivity table revisited. Application to SKINNY and AES. IACR Trans. Symmetric Cryptol. **2019**(1), 118–141 (2019). https://doi.org/10.13154/tosc.v2019.i1.118-141
26. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 244–257. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02777-2_24
27. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 128–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_5
28. Sun, L., Wang, W., Wang, M.: More accurate differential properties of LED64 and Midori64. IACR Trans. Symmetric Cryptol. **2018**(3), 93–123 (2018)
29. Sun, S., et al.: Analysis of AES, SKINNY, and others with constraint programming. In: 24th International Conference on Fast Software Encryption (2017)
30. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_9
31. Zhao, B., Dong, X., Meier, W., Jia, K., Wang, G.: Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and GIFT. Des. Codes Cryptogr. **88**(6), 1103–1126 (2020). https://doi.org/10.1007/s10623-020-00730-1