And Rijndael? Automatic Related-key Differential Analysis of Rijndael

Loïc Rouquette, David Gerault, Marine Minier, Christine Solnon

And Rijndael?

Automatic Related-key Differential Analysis of Rijndael

Loïc Rouquette[1,4], David Gérault[2], Marine Minier[3], and Christine Solnon[1]

[1] CITI, INRIA, INSA Lyon, F-69621 Villeurbanne, loic.rouquette, christine.solnon@insa-lyon.fr
[2] University of Surrey, Guildford, United Kingdom, TII, Abu Dhabi dagerault@gmail.com
[3] LORIA, Université de Lorraine, F-54000, marine.minier@loria.fr
[4] LIRIS, UMR5201 CNRS, F-69621 Villeurbanne

Abstract. Finding optimal related-key differential characteristics for a given cipher is a problem that hardly scales. For the first time, we study this problem against the 25 instances of the block cipher Rijndael, which are the little brothers of the AES. To achieve this, we adapt and improve an existing approach for the AES which is based on Constraint Programming.
The attacks presented here overpass all the previous cryptanalytic results of Rijndael. Among all our results, we obtain a 12-round (out of 13 rounds) related-key differential attack for Rijndael with a block size equal to 128 bits and a key size equal to 224 bits. We also obtain an 11-round related-key differential characteristic distinguisher for Rijndael with a block size equal to 160 bits and a key size equal to 256 bits leading to an attack on 12 rounds (out of 14 rounds).

Keywords: Related-key differential characteristics · Constraint Programming (CP) · Automatic Tools · Rijndael

1 Introduction

Cryptanalysis aims at finding non-random properties and distinguishers against classical cryptographic primitives. In particular, *differential cryptanalysis* [5] is a powerful tool against block and stream ciphers. It studies the propagation of the difference $\delta X = X \oplus X'$ between two plaintexts X and X' through the cipher E or a part of the cipher, where $\oplus$ is the exclusive OR (XOR). If the distribution of the output difference $\delta C = E_K(X) \oplus E_K(X')$ is non uniform over all the keys K, then an adversary has a distinguisher and can exploit this non-uniformity to guess part of the key K faster than exhaustive search. Most of the times, the distinguisher is constructed on a reduced number of rounds. Today, differential cryptanalysis is a public knowledge, and block ciphers have proven security bounds against differential attacks. Hence, [4] proposed to consider differences not only between the plaintexts X and X' but also between the keys K and K' to mount *related-key attacks*. In this case, the cryptanalyst is interested in finding *optimal related-key differentials*, *i.e.*, input and output differences that maximize the probability of

obtaining the output difference given the input difference. In other words, we search for δX, δK, and δC that maximize the probability that δC is equal to $E_K(X) \oplus E_{K \oplus \delta K}(X \oplus \delta X)$ for a plaintext X and a key K.

Finding an optimal related-key differential characteristic is a highly combinatorial problem that hardly scales. To simplify this problem, Knudsen [16] introduced *truncated differential characteristics* where byte or nibble differences are abstracted by single bits that indicate if there is a difference at a given position or not. Thus, the search for related-key differential characteristics is divided into two steps as done in [6,10]. In Step 1, each byte or nibble difference is abstracted by a Boolean value and the aim of this step is to find the trail which minimizes the number of active S-boxes. The goal here is to find the positions of the differences. Then, for each solution found at Step 1, Step 2 aims at instantiating each Boolean value into a valid byte or nibble difference while maximizing the overall probability of the differential characteristic crossing the S-boxes. However, some truncated differential characteristics found at Step 1 may not be valid (*i.e.*, there do not exist byte or nibble values corresponding to these difference positions). Scaling from the AES to Rijndael can only be made with a tight model for Step 1. Models described in [6,10] do not scale when increasing the block size and the key size. Only the model described in [13] has a sufficiently small number of solutions found at Step 1 to hope that the computational time will be reasonable.

In this paper, we show how to adapt the two-step solving process of [13] dedicated to the AES to compute optimal related-key differential characteristics for Rijndael [8]. Both steps are solved with Constraint Programming (CP) solvers[5]: Picat-SAT for Step 1 and Choco for Step 2. We improve the approach of [13] by better interleaving Steps 1 and 2 and exploiting bounds to stop the search sooner. We also improve the Step 2 process of [13] by decomposing the constraints associated with `MixColumns`. These improvements allow us to compute the optimal differential characteristics for all Rijndael instances but one within a reasonable amount of time.

Rijndael is a family of block ciphers (more precisely it is composed of 25 instances of the same cipher where the block size and the key size vary) originally proposed at the AES competition. But the NIST only retained as a standard its 128-bit-block version under the key sizes 128, 192 and 256 bits. Studying the security of Rijndael is interesting to enlighten the AES standardization process. Among the most interesting results, we obtain a 12-round (over 13 rounds) related-key differential distinguisher and a 12-round (over 13 rounds) attack for Rijndael with a block size equal to 128 bits and a key size equal to 224 bits. We also obtain an 11-round related-key differential distinguisher for Rijndael with a block size equal to 160 bits and a key size equal to 256 bits leading to an attack on 12 rounds out of 14.

[5] The code is available here: `https://gitlab.inria.fr/lrouquet/cp-differential-cryptanalysis/-/tree/AfricaCrypt22`.

When looking at the state of the art concerning the cryptanalysis of Rijndael, some of the results are in the single key scenario [15,27,11], [25,19,18] or in the related-key scenario [26] and none of those attacks exceeds 10 rounds.

The rest of this paper is organized as follows: in Section 2, we recall the full description of Rijndael; in Section 3, we detail the methods and our CP models. in Section 4, we sum up all the related-key differential characteristics distinguishers we obtained, give all resolution times and compare them with those of [13]; in Section 5, we present two attacks based on the most efficient distinguishers and finally, in Section 6, we conclude this paper.

2 Rijndael

Rijndael-C_{len}-K_{len}(where C_{len} is the block size and K_{len} is the key size) is a set of 25 different SPN block ciphers designed by Joan Daemen and Vincent Rijmen [9]. Each instance varies according to the block size (128, 160, 192, 224 or 256 bits) and to the key size (128, 160, 192, 224 or 256 bits) but the ciphering process is the same for all variants, except for the `ShiftRows` operation (given in Table 1) and the number of rounds (given in Table 2). It has been chosen as the new advanced encryption standard by the NIST [1] with a 128-bit block size and a key length that can be set to 128, 192 or 256 bits. The number of rounds N_r depends on the text size C_{len} and on the key size K_{len} and varies between 10 and 14. For all the versions, the current block at the input of the round i is represented by a $4 \times N_b$ matrix of bytes X_i where $N_b = (C_{len}/32)$ is the number of columns and where each byte at row j and column k is denoted by $X_i[j,k]$. The round function, repeated $N_r - 1$ times, involves four elementary mappings, all linear except the first one. Round i consists of the following transformations:

`SubBytes.` A bytewise transformation is applied on each byte of the current block using an 8-bit to 8-bit non linear S-box, denoted by `SBOX`: $SX_i[j,k] =$ `SBOX`$(X_i[j,k])$, $\forall j \in [0,3], \forall k \in [0, N_b - 1]$.

`ShiftRows.` A linear mapping rotates to the left all the rows of the current matrix SX_i. The values of the shifts denoted P_{N_b} (given in Table 1) depend on N_b: $Y_i[j,k] = SX_i[j, (P_{N_b}[j] + k) \mod N_b]$, $\forall j \in [0,3], \forall k \in [0, N_b - 1]$.

`MixColumns` is a linear multiplication of each column of the current state by a constant matrix M in the Galois field GF(2^8), that provides the corresponding column of the new state. For a given column $k \in [0, N_b - 1]$, if we denote by $\otimes$ the multiplication in GF(2^8), we have:

$$Z_i[l,k] = \sum_{j \in [0,3]} M[l,j] \otimes Y_i[j,k]$$

with M the 4×4 circulant matrix defined by its first row $= [2,3,1,1]$

`AddRoundKey` performs a bitwise XOR between the subkey RK_i of round i and the current state Z_i: $X_{i+1}[j,k] = Z_i[j,k] \oplus RK_i[j,k]$, $\forall j \in [0,3], \forall k \in [0, N_b - 1]$.

The subkeys RK_i are generated from the master key K using a `KeySchedule` algorithm composed of byte shifting, `SBOX` substitutions and XORs which is fully

Algorithm 1: Rijndael KeySchedule function

input : A key matrix K of $[4; N_k]$ bytes
output: The expanded key WK of $[4; N_b \times (N_r + 1) - 1]$ bytes
for $k \in [0, N_b]$ **and** $j \in [0, 3]$ **do**
  $\mathsf{WK}[j,\ k] \leftarrow \mathsf{K}[j,\ k]$;
for $k \in [N_b, N_b \times (N_r + 1) - 1]$ **do**
  if $k \mod N_k = 0$ **then**
    $\mathsf{WK}[0,\ k] = \mathsf{WK}[0,\ k \text{ - } N_k] \oplus \texttt{SBOX}(\mathsf{WK}[1,\ k \text{ - } 1]) \oplus \mathrm{RC}_i$;
    for $j \in [1, 3]$ **do**
      $\mathsf{WK}[j,\ k] = \mathsf{WK}[j,\ k \text{ - } N_k] \oplus \texttt{SBOX}(\mathsf{WK}[(j+1) \mod 4,\ k \text{ - } 1])$;
  else if $k > 6 \wedge k \mod N_k = 4$ **then**
    for $j \in [0, 3]$ **do**
      $\mathsf{WK}[j,\ k] = \mathsf{WK}[j,\ k \text{ - } N_k] \oplus \texttt{SBOX}(\mathsf{WK}[j,\ k \text{ - } 1])$;
  else
    for $j \in [0, 3]$ **do**
      $\mathsf{WK}[j,\ k] = \mathsf{WK}[j,\ k \text{ - } N_k] \oplus \mathsf{WK}[(j+1) \mod 4,\ k \text{ - } 1]$;
return WK

	Row			
	0	1	2	3
P_{128}	0	1	2	3
P_{160}	0	1	2	3
P_{192}	0	1	2	3
P_{224}	0	1	2	4
P_{256}	0	1	3	4

Table 1. `ShiftRows` table $P_{C_{len}}$. This table specifies the required number of byte shifts to the left according to the row number, *e.g.*, $P_{224}[3] = 4$.

C_{len}	128	160	192	224	256
K_{len}=128	10	11	12	13	14
K_{len}=160	11	11	12	13	14
K_{len}=192	12	12	12	13	14
K_{len}=224	13	13	13	13	14
K_{len}=256	14	14	14	14	14

Table 2. The number of rounds N_r of Rijndael-C_{len}-K_{len}.

described in Algorithm 1.We denote by $N_k = K_{len}/32$ the number of columns of the master key K. Note that each subkey RK_i is extracted from a main register WK in the following way: $RK_i[j,k] = WK[j,(i+1) \times N_b + k]$, $\forall j \in [0,3], \forall k \in [0, N_b - 1]$.

Those $Nr - 1$ rounds are surrounded at the top by an initial key addition with the subkey RK_0 and at the bottom by a final transformation composed by a call to the round function where the `MixColumns` operation is omitted.

Our notations are summarized below.

- X_i: the state at the beginning of round i. Note that X_i is also the state after applying the `AddRoundKey` function on the previous round X_{i-1};
- SX_i: the state of round i, after applying `SubBytes`;
- Y_i: the state of round i, after applying `ShiftRows`;
- Z_i: the state of round i, after applying `MixColumns`.

– RK_i: the subkey of round i.

3 The Solving Process

In this section, we describe how to compute the optimal related-key differential characteristics for Rijndael by adapting and improving the approach introduced in [13] for the AES. We first recall some basic principles of CP; then we describe the two-step solving process; finally, we describe the CP models associated with each of these two steps.

3.1 Constraint Programming

CP is a generic framework for solving Constraint Satisfaction Problems (CSPs), *i.e.*, finding an assignment of values to variables such that (i) each variable is assigned to a value that belongs to its domain, and (ii) a given set of constraints is satisfied. Each constraint is a relation between some variables which restricts the set of values that may be assigned simultaneously to these variables. This relation may be defined in intention, by using mathematical operators, or in extension, by listing all allowed tuples. For example, let us consider the constraint that ensures that the sum of three variables x_1, x_2, and x_3 is different from 1. This constraint may be defined in intention by using operators + and $\neq$: $x_1 + x_2 + x_3 \neq 1$ or it may be defined in extension by using a table constraint: $\langle x_1, x_2, x_3 \rangle \in \mathtt{T}_{\mathtt{sum}\neq 1}$ where $\mathtt{T}_{\mathtt{sum}\neq 1}$ is a table which enumerates every triple of values the sum of which is different from 1. For example, if the domain of x_1, x_2, and x_3 is $\{0, 1\}$, then $\mathtt{T}_{\mathtt{sum}\neq 1}$ contains the following triples: $\langle 0, 0, 0 \rangle$, $\langle 0, 1, 1 \rangle$, $\langle 1, 0, 1 \rangle$, $\langle 0, 1, 1 \rangle$, and $\langle 1, 1, 1 \rangle$.

CP may also be used to solve Constrained Optimisation Problems (COPs), *i.e.*, CSPs with an additional objective function that must be optimised. CSPs and COPs are defined by using a modelling language such as MiniZinc [20]. Then, they can be solved by CP solvers such as, Choco [21], Gecode [12], Chuffed [7], or Picat-SAT [28]. We refer the reader to [22] for more details on CP.

SAT is a special case of CSP, where all variables have boolean domains and all constraints are logical formulae (clauses). Also, MILPs (Mixed Integer Linear Programs) are special cases of COPs where all variables have numerical domains, constraints are linear inequalities, and the objective function is linear. To compute differential characteristics with SAT or MILP solvers, it is necessary to model the problem by means of logical formulae (for SAT) or linear inequalities (for MILP). In particular, the non linear `DDT` associated with an S-box must be represented by a large number of clauses (for SAT) or inequalities (for MILP), and the resulting models hardly scale [2,17,24].

When using CP, constraints do not need to be linear and `DDT`s are modelled in a straightforward way by using table constraints. In particular, [13] recently showed that CP solvers can compute optimal differential characteristics very efficiently and outperform the dedicated approaches of [6] and [10] for the AES.

Algorithm 2: Computation of optimal related-key differential characteristics for Rijndael.

Input: The size K_{len} of the key, the size C_{len} of the block and the number r of rounds
Output: An optimal related-key differential characteristic S^*
begin
  $NB_{\texttt{SBOX}} \leftarrow$ *Step1-opt*(N_k, N_b, r) where $N_k = K_{len}/32$ and $N_b = C_{len}/32$
  $UB \leftarrow 2^{-6 \cdot NB_{\texttt{SBOX}}}$
  $LB \leftarrow 0$
  while $LB < UB$ **do**
    $S_1 \leftarrow$ *Step1-next*$(N_k, N_b, r, NB_{\texttt{SBOX}})$
    if $S_1 \neq null$ **then**
      $S_2 \leftarrow$ *Step2*(N_k, N_b, r, LB, S_1)
      if $S_2 \neq null$ **then**
        $S^* \leftarrow S_2$
        $LB \leftarrow$ *probability of* S_2
    else
      $NB_{\texttt{SBOX}} \leftarrow NB_{\texttt{SBOX}} + 1$
      $UB \leftarrow 2^{-6 \cdot NB_{\texttt{SBOX}}}$
  return S^*

3.2 Two-step solving process

Finding the best related-key differential characteristic is a highly combinatorial problem. To improve the scalability of the attack, Knudsen has introduced truncated differentials [16]. The core idea is to solve the problem in two steps: In Step 1, we compute a truncated differential characteristic S_1 where each differential byte δA of the ciphering process is replaced with a boolean variable ΔA that indicates whether δA contains a difference or not (*i.e.*, $\Delta A = 0 \iff \delta A = 0$ and $\Delta A = 1 \iff \delta A \in [\![1, 2^8 - 1]\!]$); In Step 2, we instantiate S_1 into a differential characteristic S_2: for each boolean variable ΔA, if ΔA is equal to 0 in S_1, then δA is equal to 0 in S_2; otherwise δA must belong to $[\![1, 2^8 - 1]\!]$. Note that some truncated characteristics cannot be instantiated to a characteristic because some abstractions are done at Step 1.

As SBOX is the only non-linear operation, the probability of a differential characteristic only depends on the values of the differential bytes that pass through S-boxes, under the Markov assumption that rounds are independent. We denote δ_{SB} this set of bytes (including those in the key schedule), and Δ_{SB} the corresponding set of boolean variables.

A theoretical upper bound on the probability of the best differential characteristic may be computed by searching for the truncated differential characteristic which minimises the number of active S-boxes $NB_{\texttt{SBOX}} = \#\{\Delta A \mid \Delta A \in \Delta_{SB} \wedge \Delta A = 1\}$. As 2^{-6} is the maximal differential probability of the Rijndael SBOX, the best probability is upper bounded by $UB = 2^{-6 \cdot NB_{\texttt{SBOX}}}$.

UB may be larger than the actual best probability because it may be possible that the best truncated differential characteristic cannot be instantiated into a

differential characteristic, or because some non null differential bytes that go through S-boxes have a probability equal to 2^{-7} instead of 2^{-6}. Hence, the best differential characteristic is searched by alternating Step 1 and Step 2 in an iterative process which is described in Algorithm 2. First, we call *Step1-opt* to compute $NB_{\texttt{SBOX}}$, a lower bound of the number of active S-boxes in a truncated differential, and this number is used to compute a first upper bound UB on the probability. The lower bound LB on the probability is initialised to 0. Then, at each iteration of the while loop, we call *Step1-next* to compute the next truncated differential characteristic with $NB_{\texttt{SBOX}}$ active S-boxes: each time this function is called, it returns a new Step 1 solution with $NB_{\texttt{SBOX}}$ active S-boxes until they all have been computed (in this latter case, *Step1-next* returns *null*). If a new Step 1 solution S_1 has been computed, then *Step2* is called to search for the differential characteristic S_2 corresponding to S_1 whose probability is larger than LB and maximal: if such a characteristic exists, then the best characteristic S^* is updated to S_2 and LB is updated to the probability of S_2. When *Step1-next* returns *null*, all truncated characteristics with $NB_{\texttt{SBOX}}$ active S-boxes have been enumerated. In this case, we increment $NB_{\texttt{SBOX}}$ and update consequently the upper bound UB. We stop iterating when UB becomes smaller than or equal to LB: in this case S^* is equal to the optimal differential characteristic.

Algorithm 2 is different from the one used in [13]: it avoids computing useless Step 1 solutions by updating LB and UB and stopping the process when $LB \geq UB$. *Step1-opt*, *Step1-next* and *Step2* are implemented with CP solvers and the corresponding CP models are described in the next two sections.

3.3 Step 1

Both *Step1-opt* and *Step1-next* compute truncated differential characteristics: *Step1-opt* searches for the truncated characteristic that minimises $NB_{\texttt{SBOX}}$, whereas *Step1-next* searches for the next truncated characteristic given $NB_{\texttt{SBOX}}$. Both problems share the same constraints which are described in this section. *Step1-opt* is a COP which is obtained by adding the objective function: *minimise* $NB_{\texttt{SBOX}}$. *Step1-next* is a CSP which is obtained by assigning the variable $NB_{\texttt{SBOX}}$ to the optimal solution of *Step1-opt*.

A key point for Algorithm 2 to be efficient is to avoid as much as possible computing truncated characteristics which cannot be instantiated at Step 2. To this aim, we consider the model introduced in [13] which is tighter than the model of [14], *i.e.*, it computes fewer truncated characteristics that cannot be instantiated at Step 2. This model has been defined for the AES, and we show in this section how to extend it to Rijndael.

Constraints Associated with Rijndael Transformations. A basic Step 1 model for Rijndael is displayed in Model 1. Constraint (A1) relates $NB_{\texttt{SBOX}}$ with the number of active S-boxes. The other constraints are derived from Rijndael round function transformations.

$$NB_{\texttt{SBOX}} = \sum_{\Delta A \in \Delta_{SB}} \Delta A \tag{A1}$$

$$\forall i \in [0, N_r - 1], \forall j \in [0, 3], \forall k \in [0, N_b - 1], \quad \Delta SX_i[j,k] = \Delta X_i[j,k] \tag{A2}$$

$$\forall i \in [0, N_r - 2], \forall j \in [0, 3], \forall k \in [0, N_b - 1], \quad \Delta Y_i[j,k] = \Delta SX_i[j, P_{N_b}[j] + k \mod N_b] \tag{A3}$$

$$\forall i \in [0, N_r - 2], \forall k \in [0, N_b - 1], \quad \sum_{j \in [0,3]} \Delta Z_i[j,k] + \sum_{j \in [0,3]} \Delta Y_i[j,k] \in \{0, 5, 6, 7, 8\} \tag{A4}$$

$$\forall i \in [0, N_r - 2], \forall j \in [0, 3], \forall k \in [0, N_b - 1], \quad \begin{array}{c} \Delta RK_i[j,k] = \Delta WK[j, (i+1) \times N_b + k] \\ \Delta X_{i+1}[j,k] + \Delta Z_i[j,k] + \Delta RK_i[j,k] \neq 1 \end{array} \tag{A5}$$

$$\forall \omega \in [N_b, N_b \times (N_r + 1) - 1] \text{ such that } \texttt{isSbCol}(\omega), \forall j \in [0, 3], \quad \Delta SWK[j,\omega] = \Delta WK[j,\omega] \tag{A6}$$

where predicate $\texttt{isSbCol}(\omega)$ is $\omega \geq Nk - 1 \wedge \omega < N_b \times (N_r + 1) - 1 \wedge (\omega \mod N_k = N_k - 1 \vee (N_k > 6 \wedge \omega \mod N_k = 3))$

$$\forall j \in [0, 3], \forall \omega \in [N_b, N_b \times (N_r + 1) - 1], \quad \begin{array}{r} \textbf{if } \omega \mod N_k = 0 : \Delta WK[j,\omega] + \Delta WK[j,\omega - N_k] + \Delta SWK[(j+1) \mod 4, \omega - 1] \neq 1 \\ \textbf{else if } N_k > 6 \wedge k \mod N_k = 4 : \Delta WK[j,\omega] + \Delta WK[j,\omega - N_k] + \Delta SWK[j,\omega - 1] \neq 1 \\ \textbf{else} : \Delta WK[j,\omega] + \Delta WK[j,\omega - N_k] + \Delta WK[j,\omega - 1] \neq 1 \end{array} \tag{A7}$$

Model 1: Basic step 1 model for Rijndael.

`SubBytes`: As `SBOX` is bijective, there is an output difference if and only if there is an input difference. The `SubBytes` transformation at Boolean level is thus abstracted by an identity mapping ΔX_i and ΔSX_i (Constraint (A2)).

`ShiftRows`: As `ShiftRows` is just a shift at byte level, its abstraction in Step 1 is directly expressed as the equivalent shift as defined in Constraint (A3).

`MixColumns`: Multiplications of `MixColumns` cannot be mapped into the Boolean domain as the coefficients of M belong to GF(2^8). Thus, instead of encoding multiplications, we exploit the *MDS* (*Maximum Distance Separable*) property of the `MixColumns` transformation as defined in Constraint (A4).

`AddRoundKey`: It is a simple XOR between bytes of the current state Z_i and bytes of the subkey RK_i. It is modelled by constraint (A5) which prevents every triple of boolean variables involved in a same XOR from having exactly one difference. This constraint also relates variables associated with the subkey RK_i with variables associated with the expanded key WK.

`KeySchedule`: the whole `KeySchedule` process of Rijndael is described in Algorithm 1. The variables that pass through `SBOX`es are unchanged, as stated in Constraint (A6). XORs are modelled by Constraint (A7) which prevents every triple of boolean variables involved in a same XOR from having exactly one difference.

However, this simple model generates many truncated characteristics which cannot be instantiated at Step 2. This mainly comes from the fact that XORs performed by `AddRoundKey` and `KeySchedule` are modelled by constraints which

simply prevent the sum of differences to be equal to 1. Thus, we show how to refine this in the next two paragraphs.

Inference of new XOR equations from the `KeySchedule`. In Model 1, every XOR equation $\delta A \oplus \delta B \oplus \delta C = 0$ is represented by a sum constraint $\Delta A + \Delta B + \Delta C \neq 1$. This simple model is not sharp enough and generates a lot of truncated solutions that cannot be instantiated at Step 2. For example, the two XOR equations $\delta A \oplus \delta B \oplus \delta C = 0$ and $\delta B \oplus \delta C \oplus \delta D = 0$ are represented by the two sum constraints $\Delta A + \Delta B + \Delta C \neq 1$ and $\Delta B + \Delta C + \Delta D \neq 1$. When reasoning at the byte level, we easily infer that we cannot have $\delta A = 0$ and $\delta D \neq 0$, whatever the values of δB and δC are. However, when reasoning at the boolean level, the two sum constraints may be satisfied when $\Delta A = 0$ and $\Delta D = 1$ (*e.g.*, when $\Delta B = \Delta C = 1$).

To sharpen the Step 1 model and reduce the number of Step 1 solutions that cannot be instantiated at Step 2, we generate new XOR equations from the initial set of equations, by XORing them. These new equations do not change the set of solutions at the byte level. However, at the boolean level, they remove some of the truncated solutions that cannot be instantiated at Step 2. For example, when XORing the two XOR equations of our previous example, we obtain the equation $\delta A \oplus \delta D = 0$. When adding the constraint $\Delta A + \Delta D \neq 1$ to the two sum constraints, we prevent the search from generating solutions with $\Delta A = 0$ and $\Delta D = 1$.

This trick has been introduced in [13] for the AES, and we extend it to Rijndael in a straightforward way. More precisely, we consider the set of all XOR equations coming from the `KeySchedule` (this set corresponds to Constraint (A7) of Model 1). From this set, we generate all possible equations that involve no more than 4 variables by recursively XORing these equations[6]. This set of new equations is denoted ExtXOR.

Introduction of *diff* variables. As done in [13], we also introduce *diff* variables to reason on differences at the byte level: every variable $diff_{A,B}$ is a boolean variable which is true if $\delta A \neq \delta B$, and false otherwise. *diff* variables are associated with variables involved in the `KeySchedule`, in `AddRoundKey` and in `MixColumns`. This new Model is presented in Model 2.

Each variable $diff_{A,B}$ is related with ΔA and ΔB by ensuring: $diff_{A,B} + \Delta A + \Delta B \neq 1$. In other words, $diff_{A,B} = 0$ whenever $\Delta_A = \Delta_B = 0$ and $diff_{A,B} = 1$ whenever $\Delta_A \neq \Delta_B$. This corresponds to Constraints (E1) (for `KeySchedule`) and (E2) (for the `MixColumns`). These constraints also ensure symmetry, *i.e.*, $diff_{A,B} = diff_{B,A}$. Constraints (E3) and (E4) ensure transitivity (*i.e.*, if $\delta A = \delta B$ and $\delta B = \delta C$, then $\delta A = \delta C$) by constraining the sum of the corresponding *diff* variables to be different from 1.

Constraint (E5) relates *diff* variables associated with the subkey RK_i with *diff* variables associated with the expanded key WK.

[6] We do not generate equations with more than 4 variables as the number of new equations grows exponentially with respect to their size.

Constraints (E6) and (E7) are associated with the new XOR equations in ExtXOR. Two cases are considered: equations with three variables in Constraint (E6), and equations with four variables in Constraint (E7). In both cases, if at least one variable involved in the equation belongs to Δ_{SB}, then the constraint simply prevents the sum of the variables to be equal to 1. Otherwise, we exploit *diff* variables to tighten the constraint.

Finally, Constraints (E8) and (E9) ensure the MDS property of `MixColumns` on differences between pairs of columns (this constraint is partly equivalent with the linear incompatibility of [10]). Indeed, the MDS property holds between each input and output column before and after applying `MixColumns` but it also holds when XORing different columns. More precisely, if $i_1, i_2 \in [0, r-2]$ are two round numbers, and $k_1, k_2 \in [0, 3]$ are two column numbers, for every row $j \in [0, 3]$, we have

$$\begin{aligned}\delta Z_{i_1}[j][k_1] \oplus \delta Z_{i_2}[j][k_2] &= \left(\bigoplus_{l=0}^{3} M[j][l] \cdot \delta Y_{i_1}[l][k_1]\right) \oplus \left(\bigoplus_{l=0}^{3} M[j][l] \cdot \delta Y_{i_2}[l][k_2]\right) \\ &= \bigoplus_{l=0}^{3} M[j][l] \cdot (\delta Y_{i_1}[l][k_1] \oplus \delta Y_{i_2}[l][k_2])\end{aligned}$$

Therefore, the MDS property also holds for the result of the XOR of two different columns. This is modelled by Constraint (E8). This constraint removes many Step 1 solutions that cannot be instantiated at Step 2. In a rather similar way, Constraint (E9) is derived by XORing equations coming from `AddRoundKey`.

(E1)
$$\forall \omega_1, \omega_2 \in [0, N_b \times (N_r + 1) - 1], \forall j \in [0, 3] \text{ where } \omega_2 > \omega_1$$
$$\mathit{diff}_{WK[j,\omega_1], WK[j,\omega_2]} + \Delta WK[j, \omega_1] + \Delta WK[j, \omega_2] \neq 1$$
$$\mathit{diff}_{WK[j,\omega_1], WK[j,\omega_2]} = \mathit{diff}_{WK[j,\omega_2], WK[j,\omega_1]}$$

(E2)
$$\forall i_1, i_2 \in [0, N_r - 2], \forall j \in [0, 3], \forall k_1, k_2 \in [0, N_b - 1] \text{ where } (i_2, k_2) > (i_1, k_1)$$
$$\mathit{diff}_{Y_{i_1}[j,k_1], Y_{i_2}[j,k_2]} + \Delta Y_{i_1}[j, k_1] + \Delta Y_{i_2}[j, k_2] \neq 1$$
$$\mathit{diff}_{Y_{i_1}[j,k_1], Y_{i_2}[j,k_2]} = \mathit{diff}_{Y_{i_2}[j,k_2], Y_{i_1}[j,k_1]}$$
$$\mathit{diff}_{Z_{i_1}[j,k_1], Z_{i_2}[j,k_2]} + \Delta Z_{i_1}[j, k_1] + \Delta Z_{i_2}[j, k_2] \neq 1$$
$$\mathit{diff}_{Z_{i_1}[j,k_1], Z_{i_2}[j,k_2]} = \mathit{diff}_{Z_{i_2}[j,k_2], Z_{i_1}[j,k_1]}$$

(E3)
$$\forall \omega_1, \omega_2, \omega_3 \in [0, N_b \times (N_r + 1) - 1], \forall j \in [0, 3] \text{ where } \omega_3 > \omega_2 > \omega_1$$
$$\mathit{diff}_{WK[j,\omega_1], WK[j,\omega_2]} + \mathit{diff}_{WK[j,\omega_1], WK[j,\omega_3]} + \mathit{diff}_{WK[j,\omega_2], WK[j,\omega_3]} \neq 1$$

(E4)
$$\forall i_1, i_2, i_3 \in [0, N_r - 2], \forall j \in [0, 3], \forall k_1, k_2, k_3 \in [0, N_b - 1] \text{ where } (i_3, k_3) > (i_2, k_2) > (i_1, k_1)$$
$$\mathit{diff}_{Y_{i_1}[j,k_1], Y_{i_2}[j,k_2]} + \mathit{diff}_{Y_{i_2}[j,k_2], Y_{i_3}[j,k_3]} + \mathit{diff}_{Y_{i_3}[j,k_3], Y_{i_1}[j,k_1]} \neq 1$$
$$\mathit{diff}_{Z_{i_1}[j,k_1], Z_{i_2}[j,k_2]} + \mathit{diff}_{Z_{i_2}[j,k_2], Z_{i_3}[j,k_3]} + \mathit{diff}_{Z_{i_3}[j,k_3], Z_{i_1}[j,k_1]} \neq 1$$

(E5)
$$\forall i_1, i_2 \in [0, N_r - 1], \forall j \in [0, 3], \forall k_1, k_2 \in [0, N_b - 1]$$
$$\mathit{diff}_{RK_{i_1}[j,k_1], RK_{i_2}[j,k_2]} = \mathit{diff}_{WK[j,(i_1+1)\times N_b + k], WK[j,(i_2+1)\times N_b + k]}$$

(E6)
For each equation $\delta_{B1} \oplus \delta_{B2} \oplus \delta_{B3} = 0$ in ExtXOR,
$$\begin{aligned}
\text{if } \{\Delta_{B1}, \Delta_{B2}, \Delta_{B3}\} \cap \Delta_{SB} \neq \emptyset \text{ then } & \Delta_{B1} + \Delta_{B2} + \Delta_{B3} \neq 1 \\
\text{if } \{\Delta_{B1}, \Delta_{B2}\} \cap \Delta_{SB} = \emptyset \text{ then } & \mathit{diff}_{B1,B2} = \Delta_{B3} \\
\text{if } \{\Delta_{B2}, \Delta_{B3}\} \cap \Delta_{SB} = \emptyset \text{ then } & \mathit{diff}_{B2,B3} = \Delta_{B1} \\
\text{if } \{\Delta_{B1}, \Delta_{B3}\} \cap \Delta_{SB} = \emptyset \text{ then } & \mathit{diff}_{B1,B3} = \Delta_{B2}
\end{aligned}$$

(E7)
For each equation $\delta_{B1} \oplus \delta_{B2} \oplus \delta_{B3} \oplus \delta_{B4} = 0$ in ExtXOR,
$$\begin{aligned}
\text{if } \{\Delta_{B1}, \Delta_{B2}, \Delta_{B3}, \Delta_{B4}\} \cap \Delta_{SB} \neq \emptyset \text{ then } & \Delta_{B1} + \Delta_{B2} + \Delta_{B3} + \Delta_{B4} \neq 1 \\
\text{else } & \mathit{diff}_{B1,B2} = \mathit{diff}_{B3,B4} \\
& \mathit{diff}_{B1,B3} = \mathit{diff}_{B2,B4} \\
& \mathit{diff}_{B1,B4} = \mathit{diff}_{B2,B3}
\end{aligned}$$

(E8)
$$\forall i_1, i_2 \in [0, N_r - 2], \forall k_1, k_2 \in [0, N_b - 1] \text{ where } (i_2, k_2) > (i_1, k_1)$$
$$\sum_{j \in [0,3]} \mathit{diff}_{Y_{i_1}[j,k_1], Y_{i_2}[j,k_2]} + \sum_{j \in [0,3]} \mathit{diff}_{Z_{i_1}[j,k_1], Z_{i_2}[j,k_2]} \in \{0, 5, 6, 7, 8\}$$

(E9)
$$\forall i_1, i_2 \in [0, N_r - 2], \forall j \in [0, 3], \forall k_1, k_2 \in [0, N_b - 1] \text{ where } (i_2, k_2) > (i_1, k_1)$$
$$\mathit{diff}_{RK_{i_1}[j,k_1], RK_{i_2}[j,k_2]} + \mathit{diff}_{Z_{i_1}[j,k_1], Z_{i_2}[j,k_2]} + \Delta X_{i_1+1}[j, k_1] + \Delta X_{i_2+1}[j, k_2] \neq 1$$

Model 2: Additional constraints for the refined Step 1 model for Rijndael.

Incomplete Step 1 solutions. As pointed out in [13], some AES instances have a huge number of Step 1 solutions. Many of these solutions have exactly the same values for the boolean variables in Δ_{SB} (corresponding to S-boxes), and they only differ on values of other boolean variables (that do not correspond to S-boxes). For example, when the key has 192 bits and the number of rounds is equal to 10, there are 27,548 different Step 1 solutions. However, there are only 7 different assignments of values to the variables in Δ_{SB}.

As Rijndael is a generalisation of the AES, this is also true for Rijndael. Hence, as proposed in [13], we enumerate incomplete solutions such that only the variables in Δ_{SB} are assigned.

3.4 Step 2

Given a Step 1 solution (corresponding to a truncated characteristic), Step 2 aims at searching for the corresponding characteristic which has the largest probability, and such that this largest probability is larger than the best probability found so far (LB).

The CP model used to solve Step 2 is described in Model 3. For each boolean variable ΔA of Step 1, this model uses an integer variable δA to represent the corresponding differential byte. If this byte passes through an S-box (*i.e.*, $\delta A \in \delta_{SB}$), then the initial domain of δA depends on the value of ΔA in the Step 1 solution: If $\Delta A = 0$, then δA is assigned to 0; Otherwise the domain of δA is $[\![1, 255]\!]$. For each variable $\delta A \notin \delta_{SB}$, the domain of δA is $[\![0, 255]\!]$ as the associated boolean variable ΔA is not assigned in the Step 1 solution.

`SBOX`. We introduce new variables in order to represent probabilities associated with S-boxes. More precisely, for each byte $\delta A \in \delta_{SB}$ that passes through an S-box, we introduce an integer variable p_A. This variable represents the binary logarithm of the probability to observe the output difference δSA given the input difference δA. We consider the binary logarithm of the probability (instead of the probability), in order to avoid rounding errors. As the probability for Rijndael S-boxes is 0, 2^{-7}, 2^{-6}, or 1, and as we only consider values with non null probabilities, the values that may be assigned to p_A are -7, -6, and 0.

Constraint (C2) of Model 3 relates input differences, output differences and probabilities for the S-boxes applied on the plaintext, whereas Constraint (C9) relates them for the S-boxes in the `KeySchedule`. In both cases, we use a table constraint which ensures that the triple of variables belongs to a table denoted $\mathtt{T}_{\mathtt{SBOX}}$: This table contains a triple $\langle \delta_{in}, \delta_{out}, p \rangle$ for each couple of differential bytes $(\delta_{in}, \delta_{out})$ such that the `DDT` content for $(\delta_{in}, \delta_{out})$ is different from 0, and such that p is equal to: $\log_2(\frac{\#\{(X,X') \in [\![0,255]\!]^2 | (X \oplus X' = \delta_{in}) \wedge (\mathtt{SBOX}(X) \oplus \mathtt{SBOX}(X') = \delta_{out})\}}{256})$.

Objective function. We introduce an integer variable *obj* to represent the binary logarithm of the probability of the differential characteristic. The objective function is: *maximise obj*. The actual probability is computed as 2^{obj}.

$$obj = \sum_{\delta A \in \delta_{SB}} p_A \tag{C1}$$

$$obj > LB \tag{C2}$$

$\forall i \in [0, N_r - 1], \forall j \in [0, 3], \forall k \in [0, N_b - 1],$

$$\langle \delta X_i[j,k], \delta SX_i[j,k], p_{X_i}[j,k] \rangle \in \mathtt{T_{SBOX}} \tag{C3}$$

$\forall i \in [0, N_r - 1], \forall j \in [0, 3], \forall k \in [0, N_b - 1],$

$$\delta Y_i[j,k] = \delta SX_i[j, P_{N_b}[j] + k \mod N_b] \tag{C4}$$

$\forall i \in [0, N_r - 1], \forall k \in [0, N_b - 1], \forall j \in [0, 3], \forall v \in \{2, 3\}$

$$\langle \delta Y_i[j,k], v\delta Y_i[j,k] \rangle \in \mathtt{T_{MULv}} \tag{C5}$$

$\forall i \in [0, N_r - 1], \forall k \in [0, N_b - 1],$

$$\begin{array}{lll}
\langle 2\delta Y_i[0,k], 3\delta Y_i[1,k], a_i[k] \rangle \in \mathtt{T}_\oplus & \langle \delta Y_i[2,k], \delta Y_i[3,k], b_i[k] \rangle \in \mathtt{T}_\oplus & \langle a_i[k], b_i[k], \delta Z_i[0,k] \rangle \in \mathtt{T}_\oplus \\
\langle \delta Y_i[0,k], 2\delta Y_i[1,k], c_i[k] \rangle \in \mathtt{T}_\oplus & \langle 3\delta Y_i[2,k], \delta Y_i[3,k], d_i[k] \rangle \in \mathtt{T}_\oplus & \langle c_i[k], d_i[k], \delta Z_i[1,k] \rangle \in \mathtt{T}_\oplus \\
\langle \delta Y_i[0,k], \delta Y_i[1,k], e_i[k] \rangle \in \mathtt{T}_\oplus & \langle 2\delta Y_i[2,k], 3\delta Y_i[3,k], f_i[k] \rangle \in \mathtt{T}_\oplus & \langle e_i[k], f_i[k], \delta Z_i[2,k] \rangle \in \mathtt{T}_\oplus \\
\langle 3\delta Y_i[0,k], \delta Y_i[1,k], g_i[k] \rangle \in \mathtt{T}_\oplus & \langle \delta Y_i[2,k], 2\delta Y_i[3,k], h_i[k] \rangle \in \mathtt{T}_\oplus & \langle g_i[k], h_i[k], \delta Z_i[3,k] \rangle \in \mathtt{T}_\oplus
\end{array} \tag{C6}$$

$\forall i \in [0, N_r - 1], \forall k \in [0, N_b - 1], \forall j \in [0, 3], \forall v \in \{9, 11, 13, 14\}$

$$\langle \delta Z_i[j,k], v\delta Z_i[j,k] \rangle \in \mathtt{T_{MULv}} \tag{C7}$$

$\forall i \in [0, N_r - 1], \forall k \in [0, N_b - 1],$

$$\begin{array}{lll}
\langle 14\delta Z_i[0,k], 11\delta Z_i[1,k], m_i[k] \rangle \in \mathtt{T}_\oplus & \langle 13\delta Z_i[2,k], 9\delta Z_i[3,k], n_i[k] \rangle \in \mathtt{T}_\oplus & \langle m_i[k], n_i[k], \delta Y_i[0,k] \rangle \in \mathtt{T}_\oplus \\
\langle 9\delta Z_i[0,k], 14\delta Z_i[1,k], o_i[k] \rangle \in \mathtt{T}_\oplus & \langle 11\delta Z_i[2,k], 13\delta Z_i[3,k], p_i[k] \rangle \in \mathtt{T}_\oplus & \langle o_i[k], p_i[k], \delta Y_i[1,k] \rangle \in \mathtt{T}_\oplus \\
\langle 13\delta Z_i[0,k], 9\delta Z_i[1,k], q_i[k] \rangle \in \mathtt{T}_\oplus & \langle 14\delta Z_i[2,k], 11\delta Z_i[3,k], r_i[k] \rangle \in \mathtt{T}_\oplus & \langle q_i[k], r_i[k], \delta Y_i[2,k] \rangle \in \mathtt{T}_\oplus \\
\langle 11\delta Z_i[0,k], 13\delta Z_i[1,k], s_i[k] \rangle \in \mathtt{T}_\oplus & \langle 9\delta Z_i[2,k], 14\delta Z_i[3,k], t_i[k] \rangle \in \mathtt{T}_\oplus & \langle s_i[k], t_i[k], \delta Y_i[3,k] \rangle \in \mathtt{T}_\oplus
\end{array} \tag{C8}$$

$\forall i \in [0, N_r - 1], \forall j \in [0, 3], \forall k \in [0, N_b - 1],$

$$\langle \delta X_{i+1}[j,k], \delta Z_i[j,k], \delta WK[j, (i+1) \times N_b + k] \rangle \in \mathtt{T}_\oplus \tag{C9}$$

$\forall \omega \in [N_b, N_b \times (N_r + 1) - 1]$ such that $\mathtt{isSbCol}(\omega), \forall j \in [0, 3],$

$$\langle \delta WK[j,\omega], \delta SWK[j,\omega], p_{WK}[j,\omega] \rangle \in \mathtt{T_{SBOX}}$$

where predicate $\mathtt{isSbCol}(\omega)$ is $\omega \geq Nk - 1 \wedge \omega < N_r \times (N_k + 1) - 1 \wedge$
$(\omega \mod N_k = N_k - 1 \vee (N_k > 6 \wedge \omega \mod N_k = 3))$

$$\begin{array}{l}
\forall j \in [0, 3], \forall \omega \in [N_b, N_b \times (N_r + 1) - 1], \\
\qquad \text{if } \omega \mod N_k = 0 \text{ then } \langle \delta WK[j,\omega], \delta WK[j,\omega - N_k], \delta SWK[(j+1) \mod 4, \omega - 1] \rangle \in \mathtt{T}_\oplus \\
\text{elsif } N_k > 6 \wedge k \mod N_k = 4 \text{ then } \langle \delta WK[j,\omega], \delta WK[j,\omega - N_k], \delta SWK[j,\omega - 1] \rangle \in \mathtt{T}_\oplus \\
\qquad \text{else } \langle \delta WK[j,\omega], \delta WK[j,\omega - N_k], \delta WK[j,\omega - 1] \rangle \in \mathtt{T}_\oplus
\end{array} \tag{C10}$$

Model 3: Step 2 model for Rijndael.

Constraint (C1) of Model 3 ensures that *obj* is equal to the sum of every p_A such that δA is a byte that passes through an S-box. It also ensures that *obj* is strictly greater than the current lower bound *LB*.

`ShiftRows`. Constraint (C3) of Model 3 is the straightforward translation of `ShiftRows`.

`MixColumns`. Constraints (C4) to (C7) represent the `MixColumns` operation. We introduce new integer variables to represent the result of applying the Galois

Field multiplication to a byte: for each value $v \in \{2, 3\}$, and each byte $\delta Y_i[j, k]$, the variable $v\delta Y_i[j, k]$ is constrained to be equal to $v \otimes \delta Y_i[j, k]$ by the table constraint (C4), where $\mathtt{T}_{\mathtt{MULv}}$ contains every couple $(\delta A, \delta B) \in [\![0, 255]\!]^2$ such that $\delta B = v \otimes \delta A$. Then, Constraint (C5) ensures that $\delta Z_i[j, k]$ is equal to the result of XORing four bytes (corresponding to the bytes at column k of δY_i multiplied by the coefficients at row j of M). Again, this is done by using table constraints. The main novelty with respect to the model introduced in [13] for the AES is that we do not use a single table containing every tuple of five bytes such that the XOR of these bytes is equal to 0, as this table is very large (2^{40} tuples of five bytes). Instead, we introduce new variables (denoted $a_i[k]$, $b_i[k]$, etc), and we decompose the relation into three constraints such that each constraint ensures that the XOR of three variables is equal to zero. For example, the relation

$$\delta Z_i[0, k] \oplus 2\delta Y_i[0, k] \oplus 3\delta Y_i[1, k] \oplus \delta Y_i[2, k] \oplus \delta Y_i[3, k] = 0$$

is decomposed into the three following constraints:

$$\begin{aligned} \langle 2\delta Y_i[0, k], 3\delta Y_i[1, k], a_i[k] \rangle &\in \mathtt{T}_{\oplus} \\ \langle \delta Y_i[2, k], \delta Y_i[3, k], b_i[k] \rangle &\in \mathtt{T}_{\oplus} \\ \langle a_i[k], b_i[k], \delta Z_i[0, k] \rangle &\in \mathtt{T}_{\oplus} \end{aligned}$$

where $\mathtt{T}_{\oplus}$ is the table which contains every triple $(\delta A, \delta B, \delta C) \in [\![0, 255]\!]^3$ such that $\delta A \oplus \delta B \oplus \delta C = 0$. This decomposition allows us to remove some variables and simplify constraints when we know that some variables are equal to 0. For example, if $\Delta Y_i[0, k] = 0$ in the truncated characteristic, then we infer that $2\delta Y_i[0, k] = 0$ (because $2 \otimes 0 = 0$) and $a_i[k] = 3\delta Y_i[1, k]$ (because $0 \oplus \delta Y_i[1, k] \oplus a_i[k] = 0$). Hence, in this case the three previous constraints are replaced with: $\langle \delta Y_i[2, k], \delta Y_i[3, k], b_i[k] \rangle \in \mathtt{T}_{\oplus}$ and $\langle 3\delta Y_i[1, k], b_i[k], \delta Z_i[0, k] \rangle \in \mathtt{T}_{\oplus}$.

Constraints (C6) and (C7) are redundant constraints that model the $\mathtt{MixColumns}^{-1}$ operation: They do not change the solutions, but they speed up the solution process by allowing the solver to propagate in both forward (from the plaintext to the ciphertext) and backward (from the ciphertext to the plaintext) directions.

`AddRoundKey`. Constraint (C8) is a straightforward implementation of `AddRoundKey`, using table $\mathtt{T}_{\oplus}$.

`KeySchedule`. Constraints (C9) and (C10) model the `KeySchedule`. Constraint (C9) models the S-boxes of the `KeySchedule` (as described in Section 3.4). Constraint (C10) models the XORs of the `KeySchedule`, using table $\mathtt{T}_{\oplus}$. Note that we do not represent XORs with constants as they are cancelled by differential cryptanalysis.

4 Results

The Step 1 model is implemented with the MiniZinc 2.4.3 modelling language[7]. This language is accepted by many CP solvers and preliminary experiments have

[7] `https://github.com/MiniZinc/MiniZincIDE/releases/tag/2.4.3`

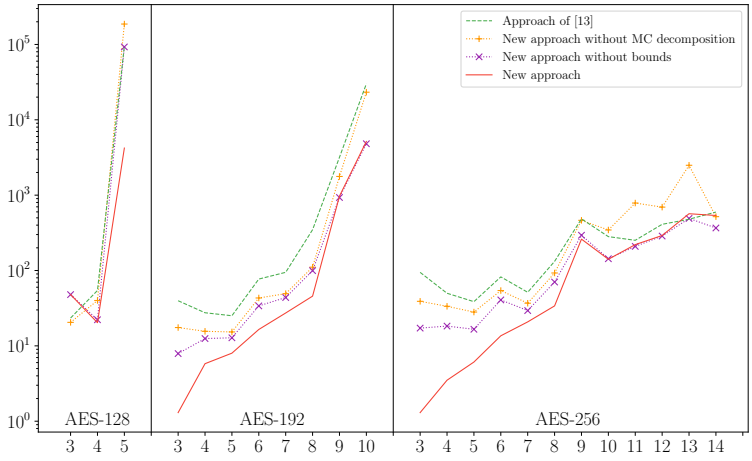

Fig. 1. Comparison of the approach of [13] (in green) with our new approach (in red), our new approach without `MixColumns` decomposition (in orange), and our new approach without exploiting bounds (in purple). Each point (x, y) corresponds to an AES instance (with $K_{len} = 128$ on the left, $K_{len} = 192$ on the middle, and $K_{len} = 256$ on the right): y gives the CPU time in seconds needed to solve it (logscale) when there are $N_r = x$ rounds.

shown us that the best performing solver is Picat-SAT[8] 2.8.6: This solver first translates the MiniZinc model into a SAT instance and then uses the Lingeling SAT solver to solve the SAT instance. The Step 2 model is implemented and solved with the CP library Choco[9] v4.10.2.

In Fig. 1, we compare solving times of the approach of [13] with those of our new approach on the AES instances in order to evaluate the interest of our two modifications, *i.e.*, (i) the interleaving of Steps 1 and 2 and the active use of LB and UB to stop the search whenever LB≥UB (see Section 3.2), and (ii) the decomposition of the `MixColumns` constraint into 3 smaller table constraints (see Section 3.4). For this experiment, all runs have been performed on a single core of an Intel Xeon CPU E3 at 3.50 GHz with 4 cores under a Linux Ubuntu 20.04.1 (Focal Fossa) using at most 16 GB of RAM. There are two instances for which our new approach needs slightly more time than the approach of [13]: AES-128 when $N_r = 3$ (48s instead of 23s) and AES-256 when $N_r = 13$ (567s instead of 479s). For the 21 remaining instances our new approach is faster and, in some cases the difference is very large, *e.g.*, 4,217s instead of 95,389s for AES-128 when $N_r = 5$, or 5,163s instead of 30,059s for AES-192 when $N_r = 10$. To

[8] `http://picat-lang.org/download/picat28_6_linux64.tar.gz`
[9] `https://github.com/chocoteam/choco-solver/releases/tag/4.10.2`

evaluate the interest of each of our two modifications separately, we also display our new approach without (ii), and our new approach without (i). In many cases, each modification improves the solution process, and the combination of these two modifications is even better. However, modification (i) deteriorates the solution process when $N_r \geq 10$ for AES-256. This comes from the fact that, for these instances, the optimal solution is strictly smaller than $2^{-6 \cdot NB_{\text{SBOX}}}$ so that the lower bound LB cannot be used to stop the search.

We give in Tables 3 and 4 the results of Algorithm 2 for every key length $K_{len} \in \{128, 160, 192, 224, 256\}$, every block size $C_{len} \in \{128, 160, 192, 224, 256\}$, and every number of rounds $N_r \in [\![3, x]\!]$ where x is the maximum number of rounds authorized (*i.e.*, the maximal number of rounds for which $NB_{\texttt{SBOX}}$ is smaller than the key length divided by 6 and the number of active S-boxes in the plaintext part is smaller than the block length divided by 6). For this experiment, all runs have been performed on a single core of an Intel Xeon E5-2630 v4 at 3.10 Ghz with 10 cores under a Linux Debian 10 (Buster) using at most 16 GB of RAM (default JVM configuration). This architecture was provided by the Grid5000 cluster [3].

Please note that there is a slight difference between the model used for Fig. 1 and the model used for Tables 3 and 4. Indeed, the model in [13] ignores the SBOXes of the last round subkey. When the key has 128 or 256 bits, this does not change anything. However, when the key has 192 bits this may change results. To allow a fair comparison with [13], we also ignore the SBOXes of the last round key in all models compared in Fig. 1. However, in Tables 3 and 4, we do consider the SBOXes of the last round subkey. Therefore, when the key has 192 bits and the text 128 bits, some probabilities may be greater than those reported in [13], *e.g.*, for the instance Rijndael-128-192 with 7 rounds the maximal probability is 2^{-84} instead of 2^{-78}.

One instance (when $C_{len} = 128$, $K_{len} = 160$, and $N_r = 8$) is still not completely solved at the time we submit the paper, after 38 days of computation. For this instance, the output value of *Step1-opt* is 23. *Step1-enum* has enumerated 7 truncated characteristics with 23 active S-boxes and none of them can be instantiated into a Step 2 characteristic. So far, we have enumerated 213 truncated characteristic with 24 active S-boxes and none of them can be instantiated into a Step 2 characteristic. Hence, for this instance the current upper bound is $UB = 2^{-150}$. We have computed 189 instances with 25 active S-boxes and 1048 instances with 26 active S-boxes and the smaller probability is $LB = 2^{-160}$.

All other instances have been solved within a reasonable amount of time: 82 are solved within 1,000s; 24 need more than 1,000s and less than 10,000s (*i.e.*, less than three hours); 10 need more than 10,000s and less than 100,000s (*i.e.*, less than 28 hours); and finally 2 need more than 28 hours and less than 3 days.

In Tables 3 and 4, o_1 is the output value of *Step1-opt* (called at line 1 of Algorithm 2), *i.e.*, the initial value of $NB_{\texttt{SBOX}}$; p is the output value of Algorithm 2, *i.e.*, the probability of the optimal related-key differential characteristic; and time is the total CPU time spent by Algorithm 2 in seconds (this time both includes the running times of Picat-SAT and of Choco). We also report the number

o_2 of active S-boxes in the optimal differential characteristic. In most cases (91 out of 122 cases), $o_1 = o_2$ and $p > 2^{-6\cdot(o_1+1)}$. In these cases, Algorithm 1 has enumerated Step 1 solutions for only one value of $NB_{\texttt{SBOX}}$, and LB became larger than or equal to UB the first time $NB_{\texttt{SBOX}}$ has been incremented.

In 17 cases (marked with ^{c} just after o_2), $o_1 = o_2$ but it has been necessary to increment $NB_{\texttt{SBOX}}$ at least once in order to check that no better characteristic can be found with more active S-boxes. For example, when $C_{len} = 128$, $K_{len} = 224$, and $r = 9$, the best differential characteristic has 22 active S-boxes and its probability is 2^{-139}. As $2^{-139} < 2^{-6\cdot 23}$, Algorithm 1 has incremented $NB_{\texttt{SBOX}}$ in order to check that it is not possible to have a larger probability (equal to 2^{-139}) with 23 active S-boxes.

In 2 cases (marked with ! just after o_2), $o_2 \geq o_1 + 1$ because none of the step 1 truncated characteristic with o_1 active S-boxes can be instantiated into a Step 2 characteristic. In these two cases, Algorithm 1 has incremented $NB_{\texttt{SBOX}}$ in order to enumerate Step 1 solutions with $o_1 + 1$ active S-boxes and find the best differential characteristic.

Finally, in 13 cases (marked with $\uparrow$ just after o_2), $o_2 \geq o_1 + 1$ because a better characteristic has been found with $o_1 + n$ active S-boxes (though at least one Step 1 solution can be instantiated into a Step 2 solution).

Results when $C_{len} = 128$ and $K_{len} \in \{128, 160, 192\}$

	$K_{len} = 128$				$K_{len} = 160$				$K_{len} = 192$			
	o_1	o_2	p	time	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	5	5	2^{-31}	13	4	4	2^{-24}	5	1	1	2^{-6}	1
$Nr = 4$	12	12	2^{-75}	31	5	5	2^{-30}	21	4	4	2^{-24}	6
$Nr = 5$	17	17	2^{-105}	8,304	10	10	2^{-60}	12	5	5	2^{-30}	8
$Nr = 6$					17	17	2^{-108}	641	10	10	2^{-60}	17
$Nr = 7$					19	19	2^{-120}	1,089	14	14	2^{-84}	46
$Nr = 8$					$23 \geq 24$!		$2^{-160} \leq p \leq 2^{-150}$	$> 10^6$	18	18	2^{-108}	83
$Nr = 9$									24	24	2^{-146}	1,800

Results when $C_{len} = 128$ and $K_{len} \in \{224, 256\}$

	$K_{len} = 224$				$K_{len} = 256$			
	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	1	1	2^{-6}	1	1	1	2^{-6}	1
$Nr = 4$	3	3	2^{-18}	3	3	3	2^{-18}	3
$Nr = 5$	6	6	2^{-36}	8	3	3	2^{-18}	5
$Nr = 6$	8	8	2^{-48}	14	5	5	2^{-30}	13
$Nr = 7$	13	13	2^{-78}	35	5	5	2^{-30}	18
$Nr = 8$	18	18	2^{-112}	1,593	10	10	2^{-60}	32
$Nr = 9$	22	22^c	2^{-139}	2,425	15	15	2^{-92}	346
$Nr = 10$	24	24^c	2^{-151}	1,834	16	16	2^{-98}	159
$Nr = 11$	27	27^c	2^{-169}	1,823	20	20	2^{-122}	330
$Nr = 12$	34	34^c	2^{-212}	9,561	20	20	2^{-122}	277
$Nr = 13$					24	24	2^{-146}	420
$Nr = 14$					24	24	2^{-146}	557

Results when $C_{len} = 160$ and $K_{len} \in \{128, 160, 192\}$

	$K_{len} = 128$				$K_{len} = 160$				$K_{len} = 192$			
	o_1	o_2	p	time	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	9	9	2^{-54}	6	5	5	2^{-30}	880	4	4	2^{-24}	4
$Nr = 4$	18	18	2^{-112}	49,501	10	10	2^{-60}	11	6	6	2^{-36}	7
$Nr = 5$					17	17	2^{-107}	621	9	9	2^{-54}	15
$Nr = 6$					21	22!	2^{-138}	36,788	15	15	2^{-90}	62
$Nr = 7$									19	19	2^{-117}	600
$Nr = 8$									23	23	2^{-141}	2,059

Results when $C_{len} = 160$ and $K_{len} \in \{224, 256\}$

	$K_{len} = 224$				$K_{len} = 256$			
	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	2	2	2^{-12}	2	1	1	2^{-6}	2
$Nr = 4$	5	5	2^{-31}	16	4	4	2^{-24}	4
$Nr = 5$	10	10	2^{-60}	18	6	6	2^{-36}	14
$Nr = 6$	15	15	2^{-90}	40	12	12	2^{-72}	42
$Nr = 7$	20	20	2^{-124}	402	15	15	2^{-93}	226
$Nr = 8$	24	24	2^{-148}	783	20	20	2^{-124}	755
$Nr = 9$	30	30^c	2^{-190}	13,081	23	23^c	2^{-146}	2,284
$Nr = 10$					27	27^c	2^{-169}	4,927
$Nr = 11$					32	32^c	2^{-204}	15,497

Table 3. Summary of the best related-key differential characteristics for Rijndael when $C_{len} \in \{128, 160\}$. The time is given in seconds.

Results when $C_{len} = 192$ and $K_{len} \in \{128, 160, 192\}$

	$K_{len} = 128$				$K_{len} = 160$				$K_{len} = 192$			
	o_1	o_2	p	time	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	9	9	2^{-54}	7	6	6	2^{-37}	20	5	5	2^{-30}	199
$Nr = 4$					15	15	2^{-94}	92	9	9	2^{-54}	15
$Nr = 5$					19	19	2^{-118}	183	14	15↑	2^{-90}	146
$Nr = 6$									19	19	2^{-117}	864
$Nr = 7$									25	25	2^{-153}	2,101

Results when $C_{len} = 192$ and $K_{len} \in \{224, 256\}$

	$K_{len} = 224$				$K_{len} = 256$			
	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	4	4	2^{-24}	7	1	1	2^{-6}	2
$Nr = 4$	8	8	2^{-48}	13	5	5	2^{-30}	10
$Nr = 5$	15	15	2^{-95}	387	12	12	2^{-72}	84
$Nr = 6$	16	17↑	2^{-103}	1,349	17	17	2^{-106}	452
$Nr = 7$	24	24^c	2^{-157}	11,908	18	18	2^{-112}	551
$Nr = 8$	32	$33↑^c$	2^{-205}	91,983	24	24	2^{-149}	951
$Nr = 9$					29	29	2^{-179}	3,397
$Nr = 10$					38	38^c	2^{-236}	88,076

Results when $C_{len} = 224$ and $K_{len} \in \{128, 160, 192\}$

	$K_{len} = 128$				$K_{len} = 160$				$K_{len} = 192$			
	o_1	o_2	p	time	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	9	9	2^{-54}	13	9	9	2^{-54}	9	6	6	2^{-37}	39
$Nr = 4$					19	19^c	2^{-122}	2,742	13	13	2^{-78}	35
$Nr = 5$									20	20	2^{-124}	1,040
$Nr = 6$									28	29↑	2^{-179}	18,632

Results when $C_{len} = 224$ and $K_{len} \in \{224, 256\}$

	$K_{len} = 224$				$K_{len} = 256$			
	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	6	6	2^{-36}	8	4	4	2^{-24}	10
$Nr = 4$	13	13	2^{-79}	121	8	8	2^{-48}	22
$Nr = 5$	16	17↑	2^{-103}	1,562	15	16↑	2^{-97}	3,267
$Nr = 6$	23	23^c	2^{-150}	1,511	18	19↑	2^{-115}	5,049
$Nr = 7$	31	31^c	2^{-196}	49,429	20	21↑	2^{-128}	1,378
$Nr = 8$					28	30↑	2^{-182}	18,377
$Nr = 9$					37	37^c	2^{-241}	210,290

Results when $C_{len} = 256$ and $K_{len} \in \{128, 160, 192\}$

	$K_{len} = 128$				$K_{len} = 160$				$K_{len} = 192$			
	o_1	o_2	p	time	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	9	9	2^{-54}	15	9	9	2^{-54}	13	9	9	2^{-54}	12
$Nr = 4$					20	21↑	2^{-130}	4,157	18	18	2^{-110}	824
$Nr = 5$									24	24	2^{-148}	4,624

Results when $C_{len} = 256$ and $K_{len} \in \{224, 256\}$

	$K_{len} = 224$				$K_{len} = 256$			
	o_1	o_2	p	time	o_1	o_2	p	time
$Nr = 3$	6	6	2^{-37}	33	5	5	2^{-30}	34
$Nr = 4$	18	18^c	2^{-115}	65,672	13	13	2^{-79}	276
$Nr = 5$	28	29!	2^{-179}	455,210	16	17↑	2^{-103}	3,084
$Nr = 6$					20	21↑	2^{-128}	2,170
$Nr = 7$					27	29↑	2^{-176}	9,237
$Nr = 8$					37	37^c	2^{-240}	191,581

Table 4. Summary of the best related-key differential characteristics for Rijndael when $C_{len} \in \{192, 224, 256\}$. The time is given in seconds.

5 Attacks

We describe in this section the best attacks we could mount based on the distinguishers found in the previous section. More precisely, two particular distinguishers have a real interest in terms of attacks. The first one is an 11-round related-key differential characteristic distinguisher on Rijndael-128-224 (presented in Table 5 that allows us to mount an attack on 12 rounds (out of 13) of this cipher. There also exists a 12-round distinguisher for Rijndael-128-224 but due to its very low probability (equal to 2^{-127}) for the data path, we do not manage to transform this distinguisher into an attack. And second, we also mount an attack on 12 rounds of Rijndael-160-256 (it has 14 rounds) based on the 11-round related-key differential characteristic distinguisher (presented in Table 6).

5.1 Attack on 12 rounds of Rijndael-128-224

First, remember that the 12th round of Rijndael-128-224 is the last round for our attack so it does not contain a `MixColumns` operation. We base our attack on the distinguisher presented in Table 5. This distinguisher has a probability equal to 2^{-169}: 2^{-103} coming from the state and 2^{-66} coming from the key.

Round	$\delta X_i = X_i \oplus X'_i$ (before `SBOX`) δSBX_i (after `SBOX`)	δRK_i	Pr(States)	Pr(Key)
$i=0$	005D005D 00A300A3 00A300A3 00FE00FE	015C005D 00A300A3 00A300A3 00FE00FE	–	–
$i=1$	01010000 00000000 00000000 00000000 1F1F0000 00000000 00000000 00000000	21210001 1F1F0000 1F1F0000 21210000	$2^{-2\times 6}$	–
2	1F1F0001 00000000 00000000 00000000 A3A3001F 00000000 00000000 00000000	5D5D0021 A3A3001F A3A3001F FEFE0021	$2^{-3\times 6}$	–
3	0000001F 00000000 00000000 00000000 000000A3 00000000 00000000 00000000	0000015C 000000A3 000000A3 000000FE	2^{-6}	–
4	00001F1F 00000000 00000000 00000000 00001F1F 00000000 00000000 00000000	01013E3E 00001F1F 00001F1F 00002121	$2^{2\times(-6)}$	2^{-6}
5	01010000 00000000 00000000 00000000 1FA30000 00000000 00000000 00000000	3E5C0001 1FA30000 1FA30000 21FE0000	2^{-6-7}	$2^{-6-3\times 7}$
6	00010001 00000000 00000000 00000000 001F001F 00000000 00000000 00000000	003E003E 001F001F 001F001F 00210021	$2^{2\times(-6)}$	$2^{-6-3\times 7}$
7	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	01010000 00000000 00000000 00000000	–	–
8	01010000 00000000 00000000 00000000 1F1F0000 00000000 00000000 00000000	3E3E0001 1F1F0000 1F1F0000 21210000	$2^{2\times(-6)}$	–
9	00000001 00000000 00000000 00000000 0000001F 00000000 00000000 00000000	0000003E 0000001F 0000001F 00000021	2^{-6}	–
10	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000101 00000000 00000000 00000000	–	–
11	00000101 00000000 00000000 00000000 00001F1F 00000000 00000000 00000000	01012121 00001F1F 00001F1F 00002121	$2^{2\times(-6)}$	2^{-6}
output	01013E3E 00001F1F 00001F1F 00002121			

Table 5. The Best related key differential characteristic we found on 11 rounds of Rijndael-128-224 with probability equal to 2^{-169}. The four words represent the four rows of the state and are given in hexadecimal notation. Note that the last round does not contain the `MixColumns` operation.

Thus, the attack process is the following one. We submit $M = 2^{103+\epsilon}$ pairs of plaintexts X and X' with the difference specified in the first line of Table 5 under the keys K and $K' = K \oplus \delta K$ with the difference specified in the first line (second column) of Table 5. Then a possible propagation of the difference is the one shown in Table 5, and we obtain the corresponding ciphertexts C and C'.

We know from Table 5 that the output of the 11th round (and the beginning of the 12th round) is of the form $\delta X_{12} = \begin{pmatrix} 01 & 01 & 1F & 1F \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$. After passing through `SubBytes` and `ShiftRows`, it becomes: $\delta SX_{12} = \begin{pmatrix} ? & ? & ? & ? \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$. From the keyschedule, the subkey difference δK_{12} will be of the form $\begin{pmatrix} 21 & A \oplus 01 & A & A \oplus 01 \\ 1F & B & B & B \\ 1F & C & C & C \\ 21 & D & D & D \end{pmatrix}$ where A, B, C and D are unknown difference. Thus the difference between C and C' will be of the form $\delta C = \begin{pmatrix} ? & ? & ? & ? \\ 1F & B & B & B \\ 1F & C & C & C \\ 21 & D & D & D \end{pmatrix}$.

So the attack works as follows:

1. We filter on the values $1F$, $1F$, and 21 at positions $(1,0)$, $(2,0)$, and $(3,0)$ in δC before the last `ShiftRows`. It remains $2^{103+\epsilon-24} = 2^{79+\epsilon}$ pairs of plaintexts/ciphertexts. Moreover, we know that the three bytes at positions $(1,1), (1,2)$ and $(1,3)$ must be equal (this remark also holds for the second and the third rows). This leads to another filter of 48 bits.
2. We guess the byte value of K_{12} at position $(0,0)$ with a cost of 2^8. Then, we decipher this byte from C and C' to check if it is equal to 01 at the input of the 12th round. Then, it filters 2^{-8} values. Moreover, the known byte at position $(0,0)$ in K_{12} gives us the difference D (due to the keyschedule construction).
3. We can guess the byte at position $(1,0)$ in K_{12} and check the value at the input of the 12th round at position $(1,0)$, by deciphering from C and C'. Then, it filters 2^{-8} values. Moreover, we can compute the difference A.
4. We can guess the three bytes at positions $(0,1)$, $(0,2)$, and $(0,3)$ in K_{12} and check the value at the input of the 12th round at the same position knowing the difference A, by deciphering from C and C'. Then, it filters 2^{-24} values.
5. We can guess the byte at position $(3,0)$ in K_{12} and check the value at the input of the 12th round at position $(3,0)$, by deciphering from C and C'. Then, it filters 2^{-8} values.
6. We can guess the byte at position $(2,0)$ in K_{12} and check the value at the input of the 12th round at position $(2,0)$, by deciphering from C and C'. Then, it filters 2^{-8} values.

Then, we have guessed 7 bytes of the subkey K_{12}, 56 bits of key, and we have filtered an equivalent of 72 bits, leading to keep $2^{103+\epsilon-72} = 2^{31+\epsilon}$ pairs of plaintexts/ciphertexts. After guessing the 7-byte difference in the subkey K_{12}, $\delta_{K_{12}}$ is fully determined. Thus, guessing the bytes of one key state could determine the bytes of the related-key state.

The related-key differential characteristic given in Table 5 has a probability to happen for the state part equals to 2^{-103}. Thus, if we use 2^{104} plaintexts/ciphertexts in the related-key differential attack on 12 rounds of Rijndael-128-224, then the right difference of the 56 bits of the last subkey will be counted at least twice on average whereas the probability that a bad key appears twice is really low (around $2^{32-72} = 2^{-40}$). The success probability computed using the results of [23] is around 97%. The time complexity of the attack is about 2^{105} encryptions and the attack succeeds if the key follows the characteristic described in Table 5. In other words, we have a set of weak keys of size $2^{224-66} = 2^{158}$.

The 168 remaining bits of the master key can be guessed through guessing more bytes in K_{11} and in K_{12} and filtering according to the remaining values in δX_{11} and the `SBox`es of the key schedule.

5.2 Attack on 12 rounds of Rijndael-160-256

Round	$\delta X_i = X_i \oplus X'_i$ (before `SBOX`) δSBX_i (after `SBOX`)	δRK_i	Pr(States)	Pr(Key)
$i = 0$	E094E0E082 7000700041 1400700041 701F9000C3	E000E0E000 7000700041 7000700041 70909000C3	–	–
$i = 1$	0094000082 0000000000 6400000000 008F000000 0041000070 0000000000 2000000000 0010000000	008282E0E0 0041007070 0041007070 00C3000090	$2^{4\times(-6)}$	2^{-6}
2	000082D000 0000000000 0000000000 0000000000 0000414100 0000000000 0000000000 0000000000	00E0828200 0000414100 0000414100 0000C3C300	$2^{2\times(-7)}$	2^{-6}
3	00E0000000 0000000000 0000000000 0000000000 0070000000 0000000000 0000000000 0000000000	82E00000E0 0070000000 0070000000 0090000000	2^{-7}	2^{-6-7}
4	82000000E0 0000000000 0000000000 0000000000 4100000070 0000000000 0000000000 0000000000	00828200E0 4100000070 4100000070 C300000090	$2^{2\times(-7)}$	–
5	8282820000 0000000000 0000000000 0000000000 7070700000 0000000000 0000000000 0000000000	E0E0000082 7070700000 7070700000 9090900000	$2^{3\times(-6)}$	$2^{3\times(-7)}$
6	0000E00082 0000000000 0000000000 0000000000 0000700070 0000000000 0000000000 0000000000	0000E000E0 0000700070 0000700070 0000900090	2^{-6-7}	–
7	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000	E082000000 0000000000 0000000000 0000000000	–	2^{-6}
8	E082000000 0000000000 0000000000 0000000000 7070000000 0000000000 0000000000 0000000000	E0E000E000 7070000000 7070000000 9090000000	2^{-6-7}	2^{-6}
9	000000E000 0000000000 0000000000 0000000000 0000007000 0000000000 0000000000 0000000000	000000E000 0000007000 0000007000 0000009000	2^{-7}	–
10	0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000	00E0828282 0000000000 0000000000 0000000000	–	2^{-6}
11	00E0828282 0000000000 0000000000 0000000000 0082707070 0000000000 0000000000 0000000000	82E0E0E000 0070707070 0070707070 00E0E0E0E0	$2^{4\times(-6)}$	2^{-6}
Output	?????????? 00F20000?? 00F20000?? 000D0000??			

Table 6. The Best related key differential characteristic we found on 11 rounds of Rijndael-160-256 with probability equal to 2^{-204}. The four words represent the four rows of the state and are given in hexadecimal notation. Note that the last round does not contain the `MixColumns` operation.

In the same way, from the related-key differential characteristic distinguisher on 11-round of Rijndael-160-256 presented in Table 6, we can easily mount a 12-round attack against Rijndael-160-256 that has 14 rounds. Note that the 12th round does not contain the `MixColumns` operation as it is the last round of our attack. The 11-round related-key differential characteristic distinguisher presented in Table 6 has a probability equal to 2^{-204}: 2^{-128} coming from the difference in the state and 2^{-76} coming from the key.

Thus, the attack process is the following one. We submit $M = 2^{128+\epsilon}$ pairs of plaintexts X and X' with the difference specified in the first line of Table 6 under the keys K and $K' = K \oplus \delta K$ with the difference specified in the first line (second column) of Table 6. Then a possible propagation of the difference is the one shown in Table 6. Then, we obtain the corresponding ciphertexts C and C'.

Then, we know from Table 6 that the output of the 11th round (and the beginning of the 12th round) is of the form $\delta X_{12} = \begin{pmatrix} 82 & FF & 0 & 0 & E0 \\ 0 & F2 & 0 & 0 & 0 \\ 0 & F2 & 0 & 0 & 0 \\ 0 & ED & 70 & 70 & 70 \end{pmatrix}$. After passing through `SubBytes` and `ShiftRows`, it becomes: $\delta SX_{12} = \begin{pmatrix} ? & ? & 0 & 0 & ? \\ ? & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ? \\ ? & ? & 0 & ? & ? \end{pmatrix}$. From the keyschedule, the subkey differences δK_{12} will be of the form $\begin{pmatrix} 82 & 0 & 82 & 0 & F \\ A & A & A & A & D \\ B & B & B & B & E \\ C & C & C & C & E0 \end{pmatrix}$ where A, B, C, D, E and F are unknown difference. Thus the difference between C and C' will be of the form $\delta C = \begin{pmatrix} ? & ? & 82 & 0 & ? \\ ? & A & A & A & D \\ B & B & B & B & ? \\ ? & ? & C & ? & ? \end{pmatrix}$.

So the attack works as follows:

1. For all the $2^{128+\epsilon}$ encrypted pairs of plaintexts/ciphertexts, we filter on the values 82 and 0 at positions $(0, 2)$ and $(0, 3)$ in δC. This filters 2^{-16} values. Then, it remains $2^{112+\epsilon}$ encrypted pairs of plaintexts/ciphertexts.
2. We guess the three bytes at positions $(1, 4)$, $(2, 4)$, and $(3, 4)$ in K_{11} for a cost of 2^{24}. This gives us the values of differences A, B and C. With those known values, we filter on δSX_{12} on the 8 positions that are equal to 0 after removing A, B, and C (positions $(1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (2, 2), (2, 3)$ and $(3, 2)$). This filters 2^{-64} values.
3. We guess 6 bytes of K_{12} (those at positions $(0, 0), (0, 1), (1, 0), (3, 0), (3, 1)$ and $(3, 3)$). We filter the corresponding 2^{-48} values on δX_{12} (before the `SBOX`es) at the same positions.
4. We guess the byte at position $(2, 3)$ in K_{12} to get one new byte in δK_{12} at position $(1, 4)$ equal to D and check if the difference is equal to 0 at position $(1, 4)$ in δX_{12}. It filters 2^{-8} values.

5. We guess the byte at position $(3,4)$ in K_{12} to filter one byte of value in δX_{12} at position $(3,4)$. We guess another byte at position $(2,4)$ in K_{12} to filter the byte value at position $(2,4)$ in δX_{12}. And finally, we guess the two bytes at positions $(1,3)$ and $(0,4)$ in K_{12} to filter the byte value at position $(0,4)$ in δX_{12}.

We have guessed a total of 112 key bits and we have filtered, in the initial step 16 bits and the equivalent of 32 bits in the second step and the last step leading to stay with $2^{80+\epsilon}$ pairs of plaintexts/ciphertexts.

The related-key differential characteristic given in Table 6 has a probability to happen for the state part equals to 2^{-128}. Thus, if we use 2^{129} plaintexts/ciphertexts in the related-key differential attack on 12 rounds of Rijndael-160-256, then the right difference of the 112 bits of the last subkey will be counted at least twice on average whereas the probability that a bad key appears twice is really low (around $2^{81-112} = 2^{-31}$). The success probability computed using the results of [23] is around 97% also. The time complexity of the attack is about 2^{130} encryptions and the attack succeeds if the key follows the characteristic described in Table 6. In other words, we have a set of weak keys of size $2^{256-76} = 2^{180}$.

The 144 remaining bits of the master key can be guessed through guessing more bytes in K_{11} and in K_{12} and filtering according the remaining values in δX_{11} and the `SBox`es of the key schedule.

6 Conclusion

In this paper, we have extended and improved the models initially proposed in [13] for the AES to the 25 instances of Rijndael. This allowed us to compute optimal related-key differential characteristics for all Rijndael instances but one (and provide upper and lower bounds for the remaining one). We sum up in Table 7 the best attacks described in this paper.

Instance	Nb rounds Nr		Time	Number of keys
Rijndael-128-224	12	13	2^{105}	2^{158}
Rijndael-160-256	12	14	2^{130}	2^{180}

Table 7. Summary of the best related-key differential attacks we found on different Rijndael instances. The last column displays the number of keys for which the attack works.

Those results are obtained using a two-step strategy that is feasible in terms of memory usage and time consumption. This strategy is modelled in MiniZinc, and it is solved by combining two solvers: Picat-SAT for Step 1 and Choco for Step 2. It essentially means that we could today automate a big part of cryptanalysis through the use of generic solvers: we have gone from the artisan age of cryptanalysis to its industrial age.

Acknowledgements

This work has been partly funded by the ANR under grant Decrypt ANR-18-CE39-0007.

References

1. FIPS 197, Advanced Encryption Standard (AES). page 51, 2001.
2. Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP modeling for (large) s-boxes to optimize probability of differential characteristics. *IACR Trans. Symmetric Cryptol.*, 2017(4):99–129, 2017.
3. Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing, 2013.
4. Eli Biham. New types of cryptoanalytic attacks using related keys (extended abstract). In *Advances in Cryptology - EUROCRYPT '93*, volume 765 of *LNCS*, pages 398–409. Springer, 1993.
5. Eli Biham and Adi Shamir. Differential cryptoanalysis of feal and n-hash. In *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *LNCS*, pages 1–16. Springer, 1991.
6. Alex Biryukov and Ivica Nikolic. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 322–344. Springer, 2010.
7. Geoffrey Chu and Peter J. Stuckey. Chuffed solver description, 2014. Available at `http://www.minizinc.org/challenge2014/description_chuffed.txt`.
8. Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael. 1999.
9. Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES - the advanced encryption standard*. Springer, Berlin; London, 2002. OCLC: 751525895.
10. Pierre-Alain Fouque, Jérémy Jean, and Thomas Peyrin. Structural evaluation of AES and chosen-key distinguisher of 9-round AES-128. In *Advances in Cryptology - CRYPTO 2013 - Part I*, volume 8042 of *LNCS*, pages 183–203. Springer, 2013.
11. Samuel Galice and Marine Minier. Improving integral attacks against rijndael-256 up to 9 rounds. In Serge Vaudenay, editor, *Progress in Cryptology - AFRICACRYPT 2008*, volume 5023 of *LNCS*, pages 1–15. Springer, 2008.
12. Gecode Team. Gecode: Generic constraint development environment, 2006. Available from `http://www.gecode.org`.
13. David Gerault, Pascal Lafourcade, Marine Minier, and Christine Solnon. Computing AES related-key differential characteristics with constraint programming. *Artificial Intelligence*, 278:103183, January 2020.
14. David Gérault, Marine Minier, and Christine Solnon. Constraint programming models for chosen key differential cryptanalysis. In *Principles and Practice of Constraint Programming - CP 2016*, volume 9892 of *LNCS*, pages 584–601. Springer, 2016.

15. Jorge Nakahara Jr. and Ivan Carlos Pavão. Impossible-differential attacks on large-block rijndael. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *Information Security, 10th International Conference, ISC 2007*, volume 4779 of *LNCS*, pages 104–117. Springer, 2007.
16. Lars R. Knudsen. Truncated and higher order differentials. In *Fast Software Encryption: Second International Workshop - FSE*, volume 1008 of *LNCS*, pages 196–211. Springer, 1995.
17. Guozhen Liu, Mohona Ghosh, and Ling Song. Security analysis of SKINNY under related-tweakey settings (long paper). *IACR Trans. Symmetric Cryptol.*, 2017(3):37–72, 2017.
18. Ya Liu, Yifan Shi, Dawu Gu, Bo Dai, Fengyu Zhao, Wei Li, Zhiqiang Liu, and Zhiqiang Zeng. Improved impossible differential cryptanalysis of large-block rijndael. *Sci. China Inf. Sci.*, 62(3):32101:1–32101:14, 2019.
19. Marine Minier. Improving impossible-differential attacks against rijndael-160 and rijndael-224. *Des. Codes Cryptogr.*, 82(1-2):117–129, 2017.
20. Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard CP modelling language. In *Principles and Practice of Constraint Programming - CP 2007*, volume 4741 of *LNCS*, pages 529–543. Springer, 2007.
21. Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016.
22. Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.
23. Ali Aydin Selçuk. On probability of success in linear and differential cryptanalysis. *J. Cryptol.*, 21(1):131–147, 2008.
24. Ling Sun, Wei Wang, and Meiqin Wang. More accurate differential properties of LED64 and midori64. *IACR Trans. Symmetric Cryptol.*, 2018(3):93–123, 2018.
25. Qingju Wang, Dawu Gu, Vincent Rijmen, Ya Liu, Jiazhe Chen, and Andrey Bogdanov. Improved impossible differential attacks on large-block rijndael. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *Information Security and Cryptology - ICISC 2012*, volume 7839 of *LNCS*, pages 126–140. Springer, 2012.
26. Qingju Wang, Zhiqiang Liu, Deniz Toz, Kerem Varici, and Dawu Gu. Related-key rectangle cryptanalysis of rijndael-160 and rijndael-192. *IET Inf. Secur.*, 9(5):266–276, 2015.
27. Lei Zhang, Wenling Wu, Je Hong Park, Bonwook Koo, and Yongjin Yeom. Improved impossible differential attacks on large-block rijndael. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *Information Security, 11th International Conference, ISC 2008*, volume 5222 of *LNCS*, pages 298–315. Springer, 2008.
28. Neng-Fa Zhou and Håkan Kjellerstrand. The picat-sat compiler. In *Practical Aspects of Declarative Languages - PADL 2016*, volume 9585 of *LNCS*, pages 48–62. Springer, 2016.