Automatic boomerang attacks search on Rijndael

Loïc Rouquette, Marine Minier, Christine Solnon

Research Article

Loïc Rouquette*, Marine Minier, and Christine Solnon

Automatic boomerang attacks search on Rijndael

https://doi.org/10.1515/jmc-2023-0027
received September 04, 2023; accepted October 07, 2023

Abstract: Boomerang attacks were introduced in 1999 by Wagner (The boomerang attack. In: Knudsen LR, editor. FSE'99. vol. 1636 of LNCS. Heidelberg: Springer; 1999. p. 156–70) as a powerful tool in differential cryptanalysis of block ciphers, especially dedicated to ciphers with good short differentials. They have been generalized to the related-key case by Biham et al. (Related-key boomerang and rectangle attacks. In: Cramer R, editor. Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22–26, 2005, Proceedings. vol. 3494 of Lecture Notes in Computer Science. Springer; 2005. p. 507–25. doi: 10.1007/11426639_30). In this article, we show how to adapt the model proposed in 2020 by Delaune et al. (Catching the fastest boomerangs application to SKINNY. IACR Trans Symm Cryptol. 2020;2020(4):104–29) for related-key boomerang attacks on the block cipher SKINNY to the Rijndael case. Rijndael is composed of 25 instances that could be seen as generalizations of the Advanced Encryption Standard. We detail our models and present the results we obtain concerning related-key boomerang attacks on Rijndael. Notably, we present a nine-round attack against Rijndael-128-160, which has 11 rounds and beats all previous cryptanalytic results against Rijdael-128-160.

Keywords: boomerang attack, constraint programming, automatic tools, Rijndael

MSC 2020: 94A60, 90C27, 90C30

1 Introduction

The boomerang attack [1] was introduced at FSE'99 as a variant of differential attacks [2]. A cipher E is seen as the decomposition of two subciphers: $E = E_1 \circ E_0$, where the differential analysis takes place in each subcipher. Boomerang attacks are efficient when the cipher E has short differentials with high probabilities. They have been generalized to the related-key case in the study by Biham et al. [3]. Recently, new insights on what exactly happens in the middle (at the junction of E_1 and E_0) have been investigated. First, in the study by Cid et al. [4], a special table named boomerang connectivity table (BCT) has been introduced for substitution-permutation networks (SPN) to compute the probability of the middle round. Second, a careful analysis of the SKINNY cipher has been provided in the study by Delaune et al. [5] to automatically take into account more possible dependencies that could happen in the middle part of the cipher considering or not related-key. The proposed search is divided into two steps: in the first step, the possible differences are modeled by Boolean variables, and this step aims at minimizing an upper bound of the probability of the truncated boomerang distinguisher; and in the second step, which takes as input the trails found at Step 1, the model aims at maximizing the overall probability considering that the active S-boxes depend on the output of the Step 1.

*** Corresponding author: Loïc Rouquette,** LORIA, Université de Lorraine, Lorraine, F-54000, France; CITI, INRIA, INSA Lyon, F-69621 Villeurbanne, France, e-mail: loic.rouquette@insa-lyon.fr, loic.rouquette@epita.fr
Marine Minier: LORIA, Université de Lorraine, Lorraine, F-54000, France, e-mail: marine.minier@loria.fr
Christine Solnon: CITI, INRIA, INSA Lyon, F-69621 Villeurbanne, France, e-mail: christine.solnon@insa-lyon.fr

Thus, in Step 1, we compute a truncated related-key boomerang S_1 where each differential byte δA of the ciphering process is replaced by a Boolean variable ΔA that indicates whether δA contains a difference or not. In Step 2, we instantiate S_1 into a related-key boomerang distinguisher. Note that some truncated boomerangs cannot be instantiated to a boomerang because some abstractions are done at Step 1.

In this article, we implement and adapt for the Rijndael case [6] the two-step solving process of [5], which was originally proposed for SKINNY to compute related-key boomerang differential characteristics and extended to the advanced encryption standard (AES) in [7]. Note that the models proposed in [7] are quite different as they include a callback search in mixed integer linear programming (MILP) and only concern the AES with probability 1 for the key part. We do not include such a callback search in the present article. Those problems are solved with constraint programming (CP): for the first step, we use Picat-SAT [8], and for the second step, Choco [9][1].

Rijndael is a family of block ciphers (more precisely, it is 25 instances of the same cipher where the block size and the key size vary) originally proposed at the AES competition. But the National Institute of Standards and Technology (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. The standardization process was completed in 2002. What can be enriched is our understanding of the security of Rijndael and, therefore, of the AES. Among the most interesting results, we obtain a nine-round (over 11 rounds) boomerang related-key differential attack for Rijndael with a block size equal to 128 bits and a key size equal to 160 bits.

When looking at the state of the art concerning the cryptanalysis of Rijndael, some of the results are in the single-key scenario [10–15] or in the related-key scenario [16]. In this article, we obtained a related-key boomerang attack on nine rounds of Rijndael-128-160 working for 2^{121} keys among 2^{160} possible keys (meaning that the probability over the key space is equal to 2^{-39}). This is the best cryptanalytic result concerning Rijndael-128-160 when to compared to the existing attacks. Note that Rijndael-128-160 has attracted only a few cryptanalytic attention until now.

The rest of this article is organized as follows: in Section 2, we recall the full description of Rijndael, what is a boomerang attack, and how those attacks are modeled in [5]; in Section 3, we detail the methods and our CP models; in Section 4, we sum up all the related-key boomerang distinguishers we obtained and present two attacks based on the most efficient distinguishers; and finally, in Section 5, we conclude this article.

2 Preliminaries

2.1 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 Daemen and Rijmen [17]. 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 and the number of rounds (given in Table 1). It has been chosen as the new AES by the NIST [18], 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 text size C_{len} and the key size K_{len} and varies between 10 and 14. For all versions, the ith round input block is represented by a $4 \times N_b$ matrix of bytes, denoted by $X[i]$, where $N_b = (C_{\text{len}}/32)$ is the number of columns. Each byte at row j and column k of this block is denoted by $X[i, j, k]$. $X[i]$ is thus 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 $X[i-1]$. The round function is repeated $N_r - 1$ times, and it is composed of one nonlinear function (`SubBytes`) and three linear functions (`ShiftRows`, `MixColumns`, and `AddRoundKey`) described below.

– `SubBytes` is a bytewise transformation that is applied to each byte of the current block using an 8-bit to 8-bit nonlinear S-box, denoted by `SBOX`. We denote $SX[i]$ the state of round i, after applying `SubBytes`, *i.e.*, $SX[i, j, k] = \text{SBOX}(X[i, j, k])\ \forall j \in [0, 3]\ \forall k \in [0, N_b - 1]$.

1 The code is available at https://gitlab.com/rloic-gitlab/boomerang-rijndael.

Table 1: The number of rounds N_r of Rijndael-C_{len}-K_{len}

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

– `ShiftRows` is a linear mapping that rotates to the left the rows of the current matrix $SX[i]$. Shift values, denoted by $P_{N_b}[j]$, depend on the number of columns N_b and the row number j and are defined by the following table:

$$P = \begin{array}{c|cccc} N_b & j=0 & j=1 & j=2 & j=3 \\ \hline 4 & 0 & 1 & 2 & 3 \\ 5 & 0 & 1 & 2 & 3 \\ 6 & 0 & 1 & 2 & 3 \\ 7 & 0 & 1 & 2 & 4 \\ 8 & 0 & 1 & 3 & 4 \end{array}.$$

We denote $Y[i]$ the state of round i after applying `ShiftRows`, *i.e.*, $Y[i,j,k] = SX[i,j,(P_{N_b}[j]+k) \bmod N_b] \ \forall j \in [0,3] \ \forall k \in [0, N_b-1]$.

– `MixColumns` is a linear multiplication of each column of the current state by a constant matrix M in the Galois field GF(2^8). We denote $Z[i]$ the state of round i after applying `MixColumns`, *i.e.*, for each row $l \in [0,3]$ and each column $k \in [0, N_b-1]$, we have:

$$Z[i,l,k] = \sum_{j \in [0,3]} M[l,j] \otimes Y[i,j,k],$$

where $\otimes$ denotes the multiplication in GF(2^8) and M is the following 4×4 circulant matrix:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}.$$

– `AddRoundKey` performs a bitwise XOR between the subkey $RK[i]$ of round i and the current state $Z[i]$: $X[i+1,j,k] = Z[i,j,k] \oplus RK[i,j,k], \forall\, j \in [0,3], \forall k \in [0, N_b-1]$.

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**
  `WK`$[j,k] \leftarrow$ `K`$[j,k]$;
for $k \in [N_b, N_b \times (N_r+1)-1]$ **do**
  if $k \bmod N_k = 0$ **then**
    WK$[0,k]$ = WK$[0, k-N_k] \oplus$ SBOX(WK$[1, k-1]) \oplus$ RC$_i$;
    for $j \in [1,3]$ **do**
      WK$[j,k]$ = WK$[j, k-N_k] \oplus$ SBOX(WK$[(j+1) \bmod 4, k-1]$);
  else if $k > 6 \wedge k \bmod N_k = 4$ **then**
    for $j \in [0,3]$ **do**
      WK$[j,k]$ = WK$[j, k-N_k] \oplus$ SBOX(WK$[j, k-1]$);
  else
    for $j \in [0,3]$ **do**
      WK$[j,k]$ = WK$[j, k-N_k] \oplus$ WK$[(j+1) \bmod 4, k-1]$;
return `WK`

The subkeys $RK[i]$ are generated from the master key K using a `KeySchedule` algorithm composed of byte shifting, `SBOX` substitutions and XORs, which are is fully described in Algorithm 1. We denote by $N_k = K_{\text{len}}/32$ the number of columns of the master key K. Each subkey $RK[i]$ is extracted from the expanded key WK computed by Algorithm 1 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.

2.2 Boomerang attacks

The boomerang attack is a variant of the differential attack that was introduced by David Wagner in 1999 [1].

Given an initial message M_1, an adversary first constructs a second message $M_2 = M_1 \oplus \alpha$. She encrypts M_1 and M_2 to obtain C_1 and C_2, respectively. Then, she adds a difference δ to those two ciphertexts to obtain $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$. Finally, she deciphers C_3 and C_4 to obtain M_3 and M_4, respectively, and she compares $M_3 \oplus M_4$ with the initial difference $\alpha = M_1 \oplus M_2$.

In their basic form, boomerang distinguishers are built by rewriting E as the composition of two subciphers (*i.e.*, $E = E_1 \circ E_0$) and by finding a good differential for each part where the overall structure is seen as a rectangle shown in Figure 1. If p and q denote the probabilities of the differentials over the upper and lower trails E_0 and E_1 (*i.e.*, $p = \Pr(\alpha \rightarrow_{E_0} \beta)$ and $q = \Pr(\delta \rightarrow_{E_1^{-1}} \gamma)$), respectively, a first approximation returns that the probability induced by Figure 1 is thus close to p^2q^2.

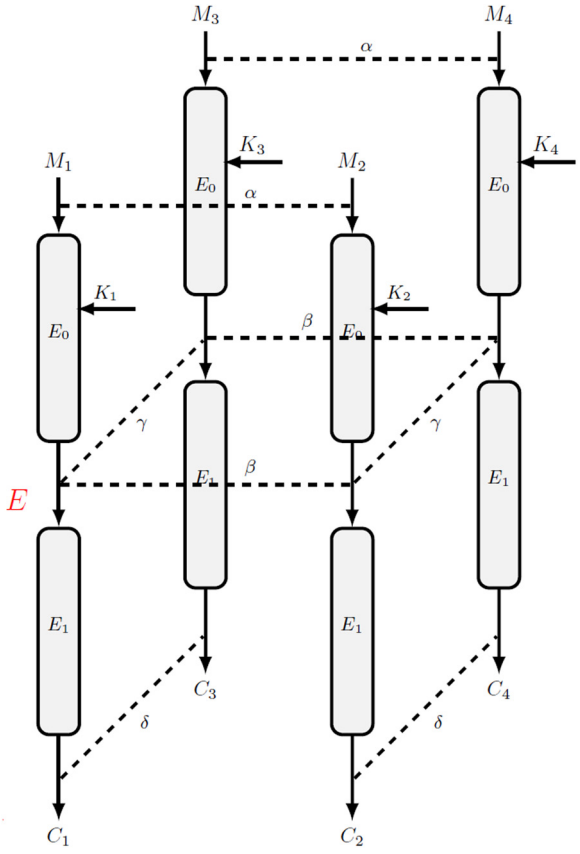

Figure 1: A related-key boomerang attack with four keys. This figure is inspired from the one of [20].

In [3], the main principle of a boomerang attack has been extended to the related-key case. In addition to the classical boomerang attack, four keys (K_1, K_2, K_3, K_4) are concerned, each one allowing to cipher one branch of the rectangle as displayed in Figure 1. The differences in the keys are chosen to be compatible with the differences between the plaintexts and are such that $K_2 = K_1 \oplus \lambda$, $K_4 = K_3 \oplus \lambda$, $K_3 = K_1 \oplus \theta$, and $K_4 = K_2 \oplus \theta$. This of course happens with a certain probability when the `KeySchedule` is nonlinear.

Recently, Cid et al. [4] have analyzed how to exactly compute the probability in the middle round using a dedicated table called the BCT for SPN ciphers. However, Song et al. [19] and Delaune et al. [5] analyzed more carefully the interactions of the boomerang on several rounds in the middle and introduce some other tables (even in the related-key setting). More precisely, the BCT is only applied on one round (the middle one), whereas the other tables are applied on several rounds (with differences that may not be fixed everywhere). A part of those tables is described in the next subsection for the SPN case.

2.3 Delaune et al.'s model

Delaune et al. [5] proposed a model divided into two steps to search for optimal boomerang distinguishers on SPN ciphers (possibly in the related-key setting). In Step 1, a MILP model searches for truncated boomerangs. Each Step 1 solution is the input of a Step 2 search that tries to instantiate the truncated boomerang with concrete differences to maximize the overall probability of the boomerang distinguisher. Our own search is organized in the same way and divided into the two same steps.

A related-key boomerang attack uses two differential trails; the first one is called the upper trail and determines α and β, the input and output differences of the distinguisher for the upper trail. The other one is called the lower trail and determines γ and δ, the input and output differences of the distinguisher for the lower trail (note that the lower trail is in the decryption direction, from ciphertext to plaintext). In the model proposed by Delaune et al. [5], the upper and lower trails are searched together on all the rounds.

For each differential byte δ_A, two variables are defined:

- a Boolean variable Δ_A, which is true if δA contains a difference, and false otherwise[2], *i.e.*, $\Delta_A = (\delta_A \neq 0)$;
- a Boolean variable free_A, which indicates whether the difference δ_A is free of conditions and can take any value with a uniform probability.

As there are two trails, these variables are duplicated: the variables used in the upper trail are denoted by $\Delta_{A_{\text{up}}}$ and $\text{free}_{A_{\text{up}}}$ and the variables used in the lower trail are denoted by $\Delta_{A_{\text{lo}}}$ and $\text{free}_{A_{\text{lo}}}$.

Constraints are added between Δ variables to model linear operators of SKINNY (`ART`, `ShiftRows`, and `MixColumns`). We do not detail these constraints as they are straightforwardly derived from the definition of linear operators. In addition, for each m-ary linear operation $(o_1, \ldots, o_m) = f(i_1, \ldots, i_m)$, a constraint is added to ensure that all output variables are free whenever any input variable is free, *i.e.*,

$$\text{free}_{i_1} \vee \ldots \vee \text{free}_{i_m} \Rightarrow \text{free}_{o_1} \wedge \ldots \wedge \text{free}_{o_m}.$$

Let us now consider the case of the nonlinear operator Sbox. For each couple of differential bytes (δ_A, δ_{SA}) such that δ_A is the input difference of an Sbox and δ_{SA} is the corresponding output difference, the constraint

$$(\Delta_{A_{\text{up}}} = \Delta_{SA_{\text{up}}}) \wedge (\Delta_{A_{\text{lo}}} = \Delta_{SA_{\text{lo}}})$$

is added to express the fact that there is an input difference iff there is an output difference, in both upper and lower trails. Then, the following constraints are defined to link together Δ and `free` variables. First, if $\delta_{A_{\text{up}}}$ is free, then $\delta_{SA_{\text{up}}}$ is also free because the upper trail follows the encryption direction. The contrary happens for the lower trail because it follows the decryption direction:

$$(\text{free}_{A_{\text{up}}} \Rightarrow \text{free}_{SA_{\text{up}}}) \wedge (\text{free}_{SA_{\text{lo}}} \Rightarrow \text{free}_{A_{\text{lo}}})$$

2 These variables are called is Active_A in [5], but we call them Δ_A to be consistent with notations introduced in [21].

In addition, if $\delta_{SA_{\text{up}}}$ is free, then $\delta_{A_{\text{up}}}$ must contain a difference. Again the lower trail is in the decryption direction.

$$(\text{free}_{SA_{\text{up}}} \Rightarrow \Delta_{A_{\text{up}}}) \wedge (\text{free}_{A_{\text{lo}}} \Rightarrow \Delta_{SA_{\text{lo}}}).$$

Each S-box probability is computed with a different table (Differential Distribution Table [DDT], DDT^2, BCT, Lower BCT [LBCT], Upper BCT [UBCT], Extended BCT [EBCT]). The right table is chosen depending on the values of the variables $\Delta_{A_{\text{up}}}$, $\Delta_{A_{\text{lo}}}$, $\text{free}_{A_{\text{up}}}$, $\text{free}_{A_{\text{lo}}}$, $\text{free}_{SA_{\text{lo}}}$, and $\text{free}_{SA_{\text{up}}}$ as defined in Model 1. We always consider a valid propagation of differences for those tables. A last constraint ensures, when a table is selected for an S-Box, that we know the required parameters to compute its transition probability.

$$(\neg\text{free}_{SA_{\text{up}}} \vee \neg\ \text{free}_{SA_{\text{lo}}}) \wedge (\neg\text{free}_{A_{\text{up}}} \vee \neg\ \text{free}_{A_{\text{lo}}})$$

The goal is to find the values that satisfy all constraints and have the maximal probability. This is done by minimizing the sum of $-\log_2(p)$ for each individual probability p. To do so, once the right table is chosen, we integrate the best possible transition probability of the chosen table to include it in the general probability computation, which is our objective function. Then, our overall probability computes the best possible probability that can be reached using the various tables, while differences are consistently propagated.

So, Step 1 outputs boomerang trails considering that the probability computation is the best one even if this best probability cannot be reached when instantiating the exact difference values. This is the role of Step 2 to instantiate the difference values and to compute the best possible probability.

$$\begin{aligned}
&\texttt{predicate}\ \ \texttt{isBCT}_{\texttt{X}}(i,j,k) = \\
&\qquad\qquad \begin{pmatrix} \Delta_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{up}}}[i,j,k] \wedge \texttt{free}_{SX_{\text{up}}}[i,j,k] \\ \wedge\ \Delta_{X_{\text{lo}}}[i,j,k] \wedge \texttt{free}_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{lo}}}[i,j,k] \end{pmatrix} \\
&\texttt{predicate}\ \ \texttt{isDDT}_{\texttt{X}}(i,j,k) = \texttt{isDDT}_{\texttt{X}}{}^{\texttt{up}}(i,j,k) \vee \texttt{isDDT}_{\texttt{X}}{}^{\texttt{lo}}(i,j,k) \\
&\texttt{predicate}\ \ \texttt{isDDT}_{\texttt{X}}{}^{\texttt{up}}(i,j,k) = \\
&\qquad \left(\Delta_{X_{\text{up}}}[i,j,k] \wedge \neg\Delta_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{up}}}[i,j,k]\right) \\
&\texttt{predicate}\ \ \texttt{isDDT}_{\texttt{X}}{}^{\texttt{lo}}(i,j,k) = \\
&\qquad \left(\neg\Delta_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{lo}}}[i,j,k] \wedge \Delta_{X_{\text{lo}}}[i,j,k]\right) \\
&\texttt{predicate}\ \ \texttt{isDDT}^2_{\texttt{X}}(i,j,k) = \texttt{isDDT}^{\texttt{2up}}_{\texttt{X}}(i,j,k) \vee \texttt{isDDT}^{\texttt{2lo}}_{\texttt{X}}(i,j,k) \\
&\texttt{predicate}\ \ \texttt{isDDT}^{\texttt{2up}}_{\texttt{X}}(i,j,k) = \\
&\qquad\qquad \begin{pmatrix} \Delta_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{up}}}[i,j,k] \\ \wedge\ \Delta_{X_{\text{lo}}}[i,j,k] \wedge \texttt{free}_{X_{\text{lo}}}[i,j,k] \wedge \texttt{free}_{SX_{\text{lo}}}[i,j,k] \end{pmatrix} \\
&\texttt{predicate}\ \ \texttt{isDDT}^{\texttt{2lo}}_{\texttt{X}}(i,j,k) = \\
&\qquad\qquad \begin{pmatrix} \Delta_{X_{\text{up}}}[i,j,k] \wedge \texttt{free}_{X_{\text{up}}}[i,j,k] \wedge \texttt{free}_{SX_{\text{up}}}[i,j,k] \\ \wedge\ \Delta_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{lo}}}[i,j,k] \end{pmatrix} \\
&\texttt{predicate}\ \ \texttt{isLBCT}_{\texttt{X}}(i,j,k) = \\
&\qquad\qquad \begin{pmatrix} \Delta_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{up}}}[i,j,k] \wedge \texttt{free}_{SX_{\text{up}}}[i,j,k] \\ \wedge\ \Delta_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{lo}}}[i,j,k] \end{pmatrix} \\
&\texttt{predicate}\ \ \texttt{isUBCT}_{\texttt{X}}(i,j,k) = \\
&\qquad\qquad \begin{pmatrix} \Delta_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{up}}}[i,j,k] \\ \wedge\ \Delta_{X_{\text{lo}}}[i,j,k] \wedge \texttt{free}_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{lo}}}[i,j,k] \end{pmatrix} \\
&\texttt{predicate}\ \ \texttt{isEBCT}_{\texttt{X}}(i,j,k) = \\
&\qquad\qquad \begin{pmatrix} \Delta_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{up}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{up}}}[i,j,k] \\ \wedge\ \Delta_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{X_{\text{lo}}}[i,j,k] \wedge \neg\texttt{free}_{SX_{\text{lo}}}[i,j,k] \end{pmatrix}
\end{aligned}$$

Model 1. Link between binary variables and tables: for each table T ∈ {BCT, DDT, DDT^2, LBCT, UBCT, EBCT}, the predicate $\text{isT}_{\text{X}}(i,j,k)$ is true iff table T must be used to link $\delta X[i,j,k]$ to $\delta SX[i,j,k]$.

3 Automatic search of related-key boomerang distinguishers on Rijndael

In this section, we detail the way we implemented the previous models to fit the case of Rijndael. As in the study by Delaune et al. [5], we divided our search into two steps: in Step 1, we search for truncated boomerang distinguishers with minimal hamming weight, whereas in Step 2, given the output Boolean differences of Step 1, we search for the instantiated boomerang distinguisher with the best probability. We describe in this section each of these two steps.

3.1 Step 1: Automatic search of related-key truncated boomerang distinguishers

As summed up in Section 3, the first step of a related-key boomerang attack may be divided into two parallel searches of related-key truncated differential characteristics (one for the upper trail and the other for the lower trail) and some glue needs to be added for the middle part using the Boolean free variables propagation. For each differential byte δA (where $A \in \{X[i,j,k], SX[i,j,k], Y[i,j,k], Z[i,j,k], RK[i,j,k] : i \in [1, N_r - 1], j \in [0,3], k \in [0, N_b - 1]\}$), we define four Boolean variables, *i.e.*, $\Delta_{A_{\text{up}}}$, $\text{free}_{A_{\text{up}}}$, $\Delta_{A_{\text{lo}}}$, and $\text{free}_{A_{\text{lo}}}$, and the meaning of these variables is the same as in the study by Delaune et al. [5].

As Rijndael also has Sboxes in the key schedule, we also introduce four Boolean variables for each differential byte $\delta_{RK}[i,j,k]$ that passes through an S-box (as defined in Algorithm 1), denoted by $\Delta_{SRK_{\text{up}}}[i,j,k]$, $\text{free}_{SRK_{\text{up}}}[i,j,k]$, $\Delta_{SRK_{\text{lo}}}[i,j,k]$, and $\text{free}_{SRK_{\text{lo}}}[i,j,k]$: these variables model the fact that there is an output difference and that this output difference is free of condition for the upper and lower trails, respectively.

Since Rijndael's KeySchedule is represented by a two-dimensional matrix, we introduce the same Δ and free variables for WK_{up}, WK_{lo}, SWK_{up}, and SWK_{lo}, which correspond to the variables RK_{up}, RK_{lo}, SRK_{up}, and SRK_{lo}. The RK and WK variables are linked by the following equations:

$$\begin{aligned}
\Delta_{RK_{trail}}[i,j,k] &= \Delta_{WK_{trail}}[j, (i+1) \times Nb + k] \\
\text{free}_{RK_{trail}}[i,j,k] &= \text{free}_{WK_{trail}}[j, (i+1) \times Nb + k] \\
\Delta_{SRK_{trail}}[i,j,k] &= \Delta_{SWK_{trail}}[j, (i+1) \times Nb + k] \\
\text{free}_{SRK_{trail}}[i,j,k] &= \text{free}_{SWK_{trail}}[j, (i+1) \times Nb + k] \\
&\text{where } trail \text{ can be either up or lo.}
\end{aligned},$$

Constraints are added between Δ variables to model Rijndael operators, and we use the same constraints as those introduced in Models 1 and 2 in the study by Rouquette et al. [22], except that these constraints are duplicated for the upper and the lower trail, respectively. For example, the constraint associated with `AddRoundKey` in Model 1 in the study by Rouquette et al. [22] is:

$$\Delta X[i+1,j,k] + \Delta Z[i,j,k] + \Delta RK[i,j,k] \neq 1.$$

In our model, this constraint becomes:

$$\begin{aligned}
\Delta X_{\text{up}}[i+1,j,k] + \Delta Z_{\text{up}}[i,j,k] + \Delta RK_{\text{up}}[i,j,k] &\neq 1 \\
\Delta X_{\text{lo}}[i+1,j,k] + \Delta Z_{\text{lo}}[i,j,k] + \Delta RK_{\text{lo}}[i,j,k] &\neq 1.
\end{aligned}$$

We do not detail these constraints here and refer the readers to Models 1 and 2 in the study by Rouquette et al. [22].

Besides these constraints, we add the new constraints defined in Model 2:

- Constraints (B1)–(B5) relate free variables together: (B1) corresponds to `AddRoundKey`, (B2) corresponds to `ShiftRows`, (B3) corresponds to `MixColumns`, and (B4) and (B5) correspond to the `KeySchedule`. Note that for each round operation (`AddRoundKey`, `ShiftRows`, and `MixColumns`), we have one constraint in the encryption direction for the upper trail, and one constraint in the decryption direction for the lower trail. For the `KeySchedule`, there are also two constraints, but they are both in the encryption direction because subkeys are all computed from the master key, in both trails.
- Constraints (B6) and (B7) define the S-Box rules that glue the two trails as done in the study by Delaune et al. [5].

– Constraint (B8) defines the objective function *obj* that must be minimized (as we consider $-\log_2$ values). There are six tables implied in the computation of the objective function: DDT, DDT^2, BCT, EBCT, LBCT, and UBCT. The predicates used to choose the correct tables are given in Model 1. These predicates are extended to *RK* variables in a straightforward way by replacing *X* with *RK*. Each predicate isT_X and isT_RK (with T ∈ {DDT, DDT^2, BCT, EBCT, LBCT, UBCT}) is multiplied by the $-\log_2$ of the maximum probability of table T, denoted by P_T.

Implementation. Our Step 1 model has been implemented in MiniZinc [23], which is a high-level and solver-independent language for modeling constraint satisfaction and optimization problems. MiniZinc models are then compiled into FlatZinc, a solver input language that is understood by a wide range of solvers (such as Choco [9], Chuffed [24], or Picat-SAT [8]). In our experiments, we have used Picat-SAT as it is the most efficient for our Step 1 problem.

ROUNDS:
$\forall i \in [0, N_r - 2], \forall j \in [0, 3], \forall k \in [0, N_b - 1],$

$$\begin{aligned} \texttt{free}_{X_{\text{up}}}[i+1,j,k] &= \texttt{free}_{Z_{\text{up}}}[i,j,k] \vee \texttt{free}_{RK_{\text{up}}}[i,j,k] \\ \texttt{free}_{Z_{\text{lo}}}[i,j,k] &= \texttt{free}_{X_{\text{lo}}}[i+1,j,k] \vee \texttt{free}_{RK_{\text{lo}}}[i,j,k] \end{aligned} \tag{B1}$$

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

$$\begin{aligned} \texttt{free}_{Y_{\text{up}}}[i,j,k] &= \texttt{free}_{SX_{\text{up}}}[i,j,P_{N_b}[j]+k \mod N_b] \\ \texttt{free}_{Y_{\text{lo}}}[i,j,k] &= \texttt{free}_{SX_{\text{lo}}}[i,j,P_{N_b}[j]+k \mod N_b] \end{aligned} \tag{B2}$$

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

$$\begin{aligned} \texttt{free}_{Z_{\text{up}}}[i,j,k] &= \texttt{free}_{Y_{\text{up}}}[i,0,k] \vee \texttt{free}_{Y_{\text{up}}}[i,1,k] \vee \texttt{free}_{Y_{\text{up}}}[i,2,k] \vee \texttt{free}_{Y_{\text{up}}}[i,3,k] \\ \texttt{free}_{Y_{\text{lo}}}[i,j,k] &= \texttt{free}_{Z_{\text{lo}}}[i,0,k] \vee \texttt{free}_{Z_{\text{lo}}}[i,1,k] \vee \texttt{free}_{Z_{\text{lo}}}[i,2,k] \vee \texttt{free}_{Z_{\text{lo}}}[i,3,k] \end{aligned} \tag{B3}$$

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

$$\begin{aligned} \texttt{free}_{RK_{\text{up}}}[i,j,k] &= \texttt{free}_{WK_{\text{up}}}[j,(i+1)\times N_b + k] \\ \texttt{free}_{RK_{\text{lo}}}[i,j,k] &= \texttt{free}_{WK_{\text{lo}}}[j,(i+1)\times N_b + k] \end{aligned} \tag{B4}$$

$\forall j \in [0, 3], \forall \omega \in [N_b, N_b \times (N_r + 1) - 1],$

$$\begin{aligned} \textbf{if } \omega \mod N_k = 0 &: \texttt{free}_{WK_{\text{up}}}[j,\omega] = \texttt{free}_{WK_{\text{up}}}[j,\omega - N_k] \vee \texttt{free}_{SWK_{\text{up}}}[(j+1)mod4, \omega - 1] \\ \textbf{else if } N_k > 6 \wedge k \mod N_k = 4 &: \texttt{free}_{WK_{\text{up}}}[j,\omega] = (\texttt{free}_{WK_{\text{up}}}[j,\omega - Nk] \vee \texttt{free}_{SWK_{\text{up}}}[j,\omega - 1]) \\ \textbf{else} &: \texttt{free}_{WK_{\text{up}}}r[j,\omega] = (\texttt{free}_{WK_{\text{up}}}[j,\omega - Nk] \vee \texttt{free}_{WK_{\text{up}}}[j,\omega - 1]) \\ \\ \textbf{if } \omega \mod N_k = 0 &: \texttt{free}_{WK_{\text{lo}}}[j,\omega] = \texttt{free}_{WK_{\text{lo}}}[j,\omega - N_k] \vee \texttt{free}_{SWK_{\text{lo}}}[(j+1)mod4, \omega - 1] \\ \textbf{else if } N_k > 6 \wedge k \mod N_k = 4 &: \texttt{free}_{WK_{\text{lo}}}[j,\omega] = (\texttt{free}_{WK_{\text{lo}}}[j,\omega - Nk] \vee \texttt{free}_{SWK_{\text{lo}}}[j,\omega - 1]) \\ \textbf{else} &: \texttt{free}_{WK_{\text{lo}}}r[j,\omega] = (\texttt{free}_{WK_{\text{lo}}}[j,\omega - Nk] \vee \texttt{free}_{WK_{\text{lo}}}[j,\omega - 1]) \end{aligned} \tag{B5}$$

OVERALL CONSTRAINTS IN BOTH DIRECTIONS:
$\forall i \in [0, N_r - 1], \forall j \in [0, 3], \forall k \in [0, N_b - 1],$

$$\begin{aligned} \texttt{free}_{SX_{\text{up}}}[i,j,k] \implies \Delta_{X_{\text{up}}}[i,j,k],\ \texttt{free}_{X_{\text{lo}}}[i,j,k] &\implies \Delta X_{\text{lo}}[i,j,k] \\ \texttt{free}_{X_{\text{up}}}[i,j,k] \implies \texttt{free}_{SX_{\text{up}}}[i,j,k],\ \texttt{free}_{SX_{\text{lo}}}[i,j,k] &\implies \texttt{free}_{X_{\text{lo}}}[i,j,k] \\ \texttt{free}_{X_{\text{up}}}[i,j,k] + \texttt{free}_{X_{\text{lo}}}[i,j,k] + \texttt{free}_{SX_{\text{up}}}[i,j,k] + \texttt{free}_{SX_{\text{lo}}}[i,j,k] &\leq 2 \end{aligned} \tag{B6}$$

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

$$\begin{aligned} \texttt{free}_{SWK_{\text{up}}}[i,\omega] \implies \Delta_{WK_{\text{up}}}[i,\omega],\ \texttt{free}_{WK_{\text{lo}}}[i,\omega] &\implies \Delta_{WK_{\text{lo}}}[i,\omega] \\ \texttt{free}_{WK_{\text{up}}}[i,\omega] \implies \texttt{free}_{SWK_{\text{up}}}[i,\omega],\ \texttt{free}_{SWK_{\text{lo}}}[i,\omega] &\implies \texttt{free}_{WK_{\text{lo}}}[i,\omega] \\ \texttt{free}_{WK_{\text{up}}}[i,\omega] + \texttt{free}_{SWK_{\text{lo}}}[i,\omega] + \texttt{free}_{WK_{\text{up}}}[i,\omega] + \texttt{free}_{SWK_{\text{lo}}}[i,\omega] &\leq 2 \end{aligned} \tag{B7}$$

where predicate $\texttt{isSBCol}(\omega) = (\omega \mod N_k = 0) \vee (N_k > 6 \wedge \omega \mod N_k = 4)$.

OBJECTIVE FUNCTION:

$$\begin{aligned} obj = \sum_{i=1}^{N_r-2}\sum_{j=0}^{3}\sum_{k=0}^{N_b-1} \begin{pmatrix} \text{P}_{DDT} \times \texttt{isDDT}_\text{X}[i,j,k] & + \text{P}_{DDT^2} \times {\texttt{isDDT}_\text{X}}^2[i,j,k] + \\ \text{P}_{BCT} \times \texttt{isBCT}_\text{X}[i,j,k] & + \text{P}_{UBCT} \times \texttt{isUBCT}_\text{X}[i,j,k] + \\ \text{P}_{EBCT} \times \texttt{isEBCT}_\text{X}[i,j,k] & \end{pmatrix} \\ + \sum_{i=1}^{N_r-1^*}\sum_{j=0}^{3}\sum_{k=0}^{N_b-1^*} \begin{pmatrix} \text{P}_{DDT} \times \texttt{isDDT}_\text{RK}[i,j,k] & + \text{P}_{DDT^2} \times {\texttt{isDDT}_\text{RK}}^2[i,j,k] + \\ \text{P}_{BCT} \times \texttt{isBCT}_\text{RK}[i,j,k] & + \text{P}_{UBCT} \times \texttt{isUBCT}_\text{RK}[i,j,k] + \\ \text{P}_{EBCT} \times \texttt{isEBCT}_\text{RK}[i,j,k] & \end{pmatrix} \end{aligned} \tag{B8}$$

* where $i \times N_b + k$ is an S-Box column in the KeySchedule

Model 2. Model linking together the `free` variables for a related-key boomerang computation.

3.2 Step 2: Instantiating the related-key truncated boomerang distinguishers

In this section, we describe how to solve Step 2, which aims at computing the maximal probability of a related-key boomerang distinguisher corresponding to a given truncated distinguisher computed in Step 1 (as explained in the previous section). We first describe the mathematical model and then show how it may be easily implemented using a CP language.

For each round $i \in [1, N_r - 1[$, each row $j \in [0, 3]$, and each column $k \in [0, N_b[$, if $\text{free}_{A_{\text{up}}}[i, j, k]$ (resp. $\text{free}_{A_{\text{lo}}}[i, j, k]$) is true in the Step 1 solution, then the corresponding differential byte $\delta_{A_{\text{up}}}[i, j, k]$ (resp. $\delta_{A_{\text{lo}}}[i, j, k]$) at Step 2 may take any value with uniform probability and is free of constraints. Hence, we do not introduce differential byte δ_A for these truncated variables. Otherwise, when $\text{free}_{A_{\text{up}}}[i, j, k]$ (resp. $\text{free}_{A_{\text{lo}}}[i, j, k]$) is false, we introduce an integer variable $\delta_{A_{\text{up}}}[i, j, k]$ (resp. $\delta_{A_{\text{lo}}}[i, j, k]$) in the Step 2 model.

For S-Box variables, the possible values of this integer variable depend on the value of its associated Δ Boolean variable (assigned at Step 1): if $\Delta_{X_{\text{up}}}[i, j, k]$ (resp. $\Delta_{X_{\text{lo}}}[i, j, k]$, $\Delta_{SX_{\text{up}}}[i, j, k]$, and $\Delta_{SX_{\text{lo}}}[i, j, k]$) is false, then $\delta_{X\text{up}}[i, j, k]$ (resp. $\delta_{X_{\text{lo}}}[i, j, k]$, $\delta_{SX_{\text{up}}}[i, j, k]$, and $\delta_{SX_{\text{lo}}}[i, j, k]$) is assigned to 0; otherwise, its set of possible values is $[1, 256[$. The same operation is done for the $\delta_{RK_{\text{up}}}$, $\delta_{RK_{\text{lo}}}$, $\delta_{SRK_{\text{up}}}$, and $\delta_{SRK_{\text{lo}}}$ variables.

Considering that ShiftRows, MixColumns, and AddRoundKey are linear functions, it is possible to infer the free state of the $\delta_{Y_{\text{up}}}$, $\delta_{Y_{\text{lo}}}$, $\delta_{Z_{\text{up}}}$, and $\delta_{Z_{\text{lo}}}$ variables from the free state of $\delta_{X_{\text{up}}}$, $\delta_{X_{\text{lo}}}$, $\delta_{SX_{\text{up}}}$, $\delta_{SX_{\text{lo}}}$, $\delta_{RK_{\text{up}}}$, $\delta_{RK_{\text{lo}}}$, $\delta_{SRK_{\text{up}}}$, and $\delta_{SRK_{\text{lo}}}$ variables. Hence, the $\delta_{Y_{\text{up}}}$, $\delta_{Y_{\text{lo}}}$, $\delta_{Z_{\text{up}}}$, and $\delta_{Z_{\text{lo}}}$ differential variables are only introduced in Step 2 when they are not free.

Finally, we introduce integer variables that represent $-\log_2$ probabilities associated with S-boxes. For each round $i \in [0, N_r - 1]$, each row $j \in [0, 3]$, and each column $k \in [0, N_b[$, we define an integer variable $p[i, j, k]$ that corresponds to the $-\log_2$ probability of crossing both the upper trail S-box (from $\delta_{\text{X}_{\text{up}}}[i, j, k]$ to $\delta_{\text{SX}_{\text{up}}}[i, j, k]$) and the lower trail S-box (from $\delta_{\text{SX}_{\text{lo}}}[i, j, k]$ to $\delta_{\text{X}_{\text{lo}}}[i, j, k]$). The values of Δ and `free` Boolean variables computed at Step 1 are used to determine which constraint must be used to link $p[i, j, k]$ variables with their corresponding differential variables as defined in Model 3. The same operation can be applied to the S-Box columns of the `KeySchedule` by introducing $p_{RK}[i, j, k]$ variables. $p_{RK}[i, j, k]$ is the probability for the Round Key byte at round i, row j, and column k to pass its S-Box. Knowing that not all the Round Key columns go through an S-Box, we only introduce a $p_{RK}[i, j, k]$ variable when necessary.

$$
\begin{aligned}
\texttt{isDDT}_\texttt{X}\texttt{up}(i,j,k) &\implies (\delta_{X_{\text{up}}}[i,j,k], \delta_{SX_{\text{up}}}[i,j,k], p[i,j,k]) \in T_{DDT}\\
\texttt{isDDT}_\texttt{X}\texttt{lo}(i,j,k) &\implies (\delta_{X_{\text{lo}}}[i,j,k], \delta_{SX_{\text{lo}}}[i,j,k], p[i,j,k]) \in T_{DDT}\\
\texttt{isDDT}^2_\texttt{X}\texttt{lo}(i,j,k) &\implies (\delta_{X_{\text{up}}}[i,j,k], \delta_{SX_{\text{up}}}[i,j,k], p[i,j,k]) \in T_{DDT^2}\\
\texttt{isDDT}^2_\texttt{X}\texttt{up}(i,j,k) &\implies (\delta_{X_{\text{lo}}}[i,j,k], \delta_{SX_{\text{lo}}}[i,j,k], p[i,j,k]) \in T_{DDT^2}\\
\texttt{isBCT}_\texttt{X}(i,j,k) &\implies (\delta_{X_{\text{up}}}[i,j,k], \delta_{SX_{\text{lo}}}[i,j,k], p[i,j,k]) \in T_{BCT}\\
\texttt{isLBCT}_\texttt{X}(i,j,k) &\implies (\delta_{X_{\text{up}}}[i,j,k], \delta_{X_{\text{lo}}}[i,j,k], \delta_{SX_{\text{lo}}}[i,j,k], p[i,j,k]) \in T_{LBCT}\\
\texttt{isUBCT}_\texttt{X}(i,j,k) &\implies (\delta_{X_{\text{up}}}[i,j,k], \delta_{SX_{\text{up}}}[i,j,k], \delta_{SX_{\text{lo}}}[i,j,k], p[i,j,k]) \in T_{UBCT}\\
\texttt{isEBCT}_\texttt{X}(i,j,k) &\implies\\
(\delta_{X_{\text{up}}}[i,j,k], \delta_{SX_{\text{up}}}[i,j,k], \delta_{X_{\text{lo}}}[i,j,k], \delta_{SX_{\text{lo}}}[i,j,k], p[i,j,k]) &\in T_{EBCT}
\end{aligned}
$$

Model 3. Constraints that relate $p[i, j, k]$ integer variables with differential variables (using predicates defined in Model 1). For each table $t \in \{DDT^2, DDT, BCT, UBCT, EBCT, LBCT\}$, T_t denotes the set of all tuples with a nonnull probability. For example, T_{DDT} contains all triples $(\delta_{\text{in}}, \delta_{\text{out}}, p)$ such that δ_{in} and δ_{out} belong to $[0, \cdots, 255]$ and $p = -\log_2(DDT(\delta_{\text{in}}, \delta_{\text{out}}))$, $p \neq 0$.

δ_X, δ_{SX}, δ_Y, δ_Z, δ_K, and δ_{RK} variables are constrained with respect to `SubBytes`, `ShiftRows`, `MixColumns`, `AddRoundKey`, and `KeySchedule` as described in Model 3 of [22], using table constraints. However, this model is duplicated for the upper and lower trails, respectively.

The objective function is then the sum of all $p[i, j, k]$ and $p_{RK}[i, j, k]$ integer variables, and the goal is to minimize this sum.

Implementation. Our Step 2 model widely uses table constraints, which are constraints of the form $(x_1, \ldots, x_n) \in T$, where x_1, ..., x_n are integer variables and T is a set of allowed tuples (of arity n). Table constraints are one of the main advantages of using CP because those tables can be defined on large alphabet and are directly handled by CP solvers.

This model has been implemented with the Choco CP library version 4.10.6 [9].

3.3 Combining the two steps

Step 1 is in fact composed of two different sub-problems: the first one, *Step1-Opt*, searches for the best possible value of *obj* (denoted by *obj**), and the second one, *Step1-Next*, searches for Step 1 solutions with the *obj* value fixed to *obj**. As there are usually many Step 1 solutions, we do not compute all of them at once, but we enumerate them one at a time, and for each enumerated Step 1 solution, Step 2 is performed to search for the best related-key differential characteristic corresponding to it, as done in the study by Rouquette et al. [22]. This search that interleaves *Step1-next* and Step 2 is iterated until finding the optimal related-key differential characteristic, or detecting that the optimal probability exceeds the block or key exhaustive search according to the Rijndael instance. Note that it may be possible that the optimal related-key differential characteristic has more than *obj** active S-boxes (either because there is no Step 2 solution with *obj** active S-boxes, or because it is possible to have a larger probability with more active S-boxes). Hence, the interleaved process increases *obj** until proving optimality of the best found differential characteristic.

Note also that Delaune et al. [5], in their original article, also proposed a way to compute the clusters induced. One of the main differences between Rijndael and SKINNY relies on the fact that the linear part of SKINNY is composed of XOR, whereas the one of Rijndael includes multiplication in a finite field. As stated in [25,26], it is out of computational reach to compute such clusters for the AES and thus Rijndael. So, due to the very high computational cost of our method without the cluster computation, we do not include any cluster in our approach.

4 Attacks

4.1 From the distinguisher to the attack

Once an efficient related-key boomerang distinguisher between rounds 1 and $N_r - 2$ is found, there exist several techniques to extend it to a key recovery attack (see, e.g., [27] for a complete survey). We focus here on the method proposed by Zhao et al. [28] even if it concerns algorithms with linear key schedule but it could be adapted for algorithms with nonlinear key schedule. Thus, we will apply their techniques to recover master key bits in round 0 and round $Nr - 1$ that do not contain `MixColumns` operation. Due to the very high diffusion of Rijndael, we do not investigate to add more rounds at the beginning or at the end of the cipher. So, let us first introduce the technique described by Zhao et al. [28].

The parameters on which the complexities of an attack are computed are the following ones:

– The distinguisher E_d with N_d rounds is placed in the middle of the attack. The input difference of the distinguisher is α, whereas the output difference is δ.
– We add N_a rounds E_a at the beginning and N_f rounds E_f at the end.

– At the beginning, in the deciphering direction, for N_a rounds, the difference α is extended backward and with probability 1 to a truncated difference α', with r_a possibly active bits and $n - r_a$ inactive bits.
– At the end, in the ciphering direction, for N_f rounds, the difference δ is extended with probability 1 to a truncated difference δ', with r_f possibly active bits and $n - r_f$ inactive bits.

Then, the attack could work on $N_a + N_d + N_f$ rounds by guessing some key materials appearing before and after the distinguisher and by counting how many times the distinguishing property happens. The correct key bits have the higher counters. In the following, the guessed key bits at the beginning are denoted by m_a and the guessed key bits at the end are denoted by m_f.

The attack of [28] works as follows where s is the expected number of right pairs:

(1) Build $y = \sqrt{s} \cdot 2^{n/2-r_a}/\sqrt{p^2q^2r}$ structures of 2^{r_a} plaintexts each, and store them with their associated plaintexts.
(2) For each possible value of the m_a key bits,
   (a) initialize 2^{m_f} key counters;
   (b) partially encrypt each plaintext M_1 of each structure using the guessed m_a key bits up to the beginning of E_d. Add α to the computed value and decrypt it up to the plaintext, to obtain M_2. Construct the set S (of size $y \cdot 2^{r_a}$) given by $S = \{(M_1, C_1, M_2, C_2)$ such that $E_a(M_1, K_1) \oplus E_a(M_2, K_2) = \alpha\}$;
   (c) insert S into a hash table H indexed by the $n - r_f$ bits that are inactive in δ', each collision defines a quartet (C_1, C_2, C_3, C_4);
   (d) use these quartets to determine the correct m_f key bits. The time complexity of this stage is denoted by ε.

The time complexity of the attack is dominated by stage 2.(b) or 2.(d). The complexity, given in the number of encryptions, for stage 2.(b) is

$$2^{m_a+r_a} \cdot y \cdot \mu = 2^{m_a+n/2} \cdot \sqrt{s} \cdot \frac{1}{\sqrt{p^2q^2r}} \cdot \mu,$$

whereas the complexity of stage 2.(d) is

$$s \cdot 2^{m_a-n+2r_f}/(p^2q^2r) \cdot \varepsilon.$$

Since stage 2.(b) does partial encryptions over E_b, μ can be approximated by $\frac{N_a}{N_a+N_d+N_f}$, while ε corresponds to the cost of gradually decrypting rounds to check the validity of a key guess and could be approximated by $\frac{1}{s}$.

Then, the success probability P_s of the related-key boomerang attack could be approximated by the success probability of a differential attack given in [29]:

$$P_s = \Phi\left(\frac{\sqrt{sS_N} - \Phi^{-1}(1 - 2^{-h})}{\sqrt{S_N + 1}}\right),$$

where S_N is the signal-to-noise ratio, so is equal to $p^2q^2r/2^{-n}$ and h is the advantage, h is typically equal to 80 (bits) or higher.

To adapt this attack to the case of a nonlinear key-schedule, one has just to compute the correct quartet of keys before the attack. Then, the probability to have a correct quartet only depends on the probability of the S-boxes induced by the `KeySchedule` and could be directly computed from the distinguisher. The probability of the distinguisher is thus computed without the probability appearing in the `KeySchedule`. Then, it may be considered as a weak key attack because all the keys could not provide a right quartet. If we denote by $P_{\text{KeySchedule}}$ the probability of the `KeySchedule` and by P_{E_d} the probability of the encryption path, then, the cost of this stage is about $4/P_{\text{KeySchedule}}$.

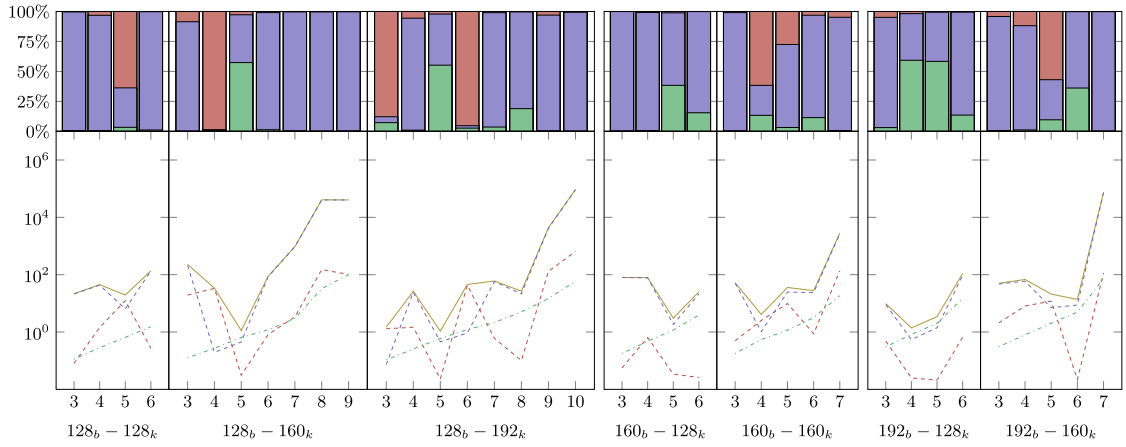

Figure 2: Computation times for instances X_b Y_k, where X is the block length and Y is the key length. The upper part of the chart represents the computation time proportion (in percentage of the total computation time) between Step-1-Opt (▬), Step-1-Enum (▬), and Step2 (▬). In the lower part, the chart represents the computation time (in seconds) for each step: Step1-Opt (-·-), Step1-Enum (– –), and Step2 (– –) and the cumulative total time (——).

4.2 Results

The Step 1 model is implemented in MiniZinc and run on Picat [8], which uses the Lingeling solver [30]. The Step 2 is implemented in Java, using the Choco library version 4.10.6 [9]. We choose the Picat solver to solve the Step 1 as it is a SAT solver, so is especially suited to problems on Boolean formulae. Previous works like [31] have shown that Picat has good performances on multiple Step 1 models. Since the Step 2 contains a lot of table constraints, it appears that CP solvers are more adapted.

All experiments are run on a virtual machine `Ubuntu 18.04.5 LTS x86_64` with an `Intel Xeon Gold 5118` processor and `32 Gio` of RAM. The requirements are : `Java 10.0.12 OpenJDK`, `Gradle 6.8`, `MiniZinc 2.5.5`, `Picat 3.1.2`, and `Choco 4.10.6`. Each instance is run on a single thread.

We put a time out of 6 months. After, those 6 months, the results we obtained are summed up in Table A1 for both Step 1 and Step 2 computations for the related-key boomerang distinguisher on Rijndael.

Even if our models could not reach the largest instances of Rijndael, we obtain the following results:

- Rijndael-128-160 with seven rounds, best probability: 2^{-95}. The version with eight rounds is not reachable.
- Rijndael-160-128 with four rounds, best probability: 2^{-18}. Rijndeal-160-128 with five rounds is not reachable.
- Rijndael-192-160 with five rounds, best probability: 2^{-73}. The version with six rounds is not reachable.

Moreover, we have those partial results (only Step 1 has finished to run) for:

- Rijndael-160-160 for six rounds with an upper bound equal to 2^{-118};
- Rijndael-160-192 for seven rounds with an upper bound equal to 2^{-102};
- Rijndael-160-224 for eight rounds with an upper bound equal to 2^{-114};
- Rijndael-160-256 for nine rounds with an upper bound equal to 2^{-120};
- Rijndael-192-128 for four rounds with an upper bound equal to 2^{-36};
- Rijndael-192-192 for six rounds with an upper bound equal to 2^{-84}.

The Step 2 for each instance has taken between 8 and 15 days. For example, for Rijndael-192-160 for six rounds, the computation for Step 2 has taken 8 days. We tested the validity of our models by experimentally checking the boomerang distinguisher on four rounds of Rijndael-128-128, the AES with a 128-bit key. We also verified that our models give us the classical bounds for the AES and for all the key lengths (128/192/256 bits) [32].

Figure 2 displays some statistics about computation times. The upper part of the figure represents which step over the three is the most time consuming, while the lower part of the figure represents the computation time. We can see that the Step 1 (Step1-Opt + Step1-Enum) step is the most time-consuming in general, expected

for six (over 37) instances. Moreover, we see that the instances where Step 2 is the most time-consuming are not among the most difficult instances. Hence, improvements should target Step 1 modeling and resolution to improve the overall performances.

4.3 Attack on nine rounds of Rijndael-128-160

The nine rounds attack of Rijndael-128-160 is presented in Figure A2 in Appendix B. The distinguisher works for rounds 1 to 8 and has a probability of 2^{-56} for the encryption part and of 2^{-39} for the `KeySchedule`.

Thus, the attack implies the following parameters: $N_a = 1$, $r_a = 32$, $n - r_a = 96$, $N_f = 1$, $r_f = 32$, $n - r_f = 96$, $m_a = 32$, $m_f = 32$, $P_{E_d} = 2^{-56}$, and $P_{\text{KeySchedule}} = 2^{-39}$. Thus, applying the previous attack, with $s = 4$, we need to cipher 2^{61} structures of 2^{32} plaintexts. The complexity of the attack is dominated by stage 2.(b) and is equal to 2^{122} encryptions. The success probability of the attack is equal to 97.67%.

The probability that a right quartet of keys is found is equal to 2^{39}, and the complexity to find such a quartet is equal to 2^{41} encryptions. Another way to say that is that the number of keys that work for our attack is equal to $2^{160-39} = 2^{121}$.

5 Conclusion

In this article, we have presented the related-key boomerang distinguishers and the related-key boomerang attacks we obtained for some of the 25 instances of the block cipher Rijndael. Among our most significant results, we obtained a nine-round attack on Rijndael-128-160, which has 11 rounds.

However, the computational costs of our models are prohibitive for the largest Rijndael instances. So, we plan to try to improve those models and notably the way the Step 1 is computed to try to reach the missing instances.

Acknowledgements: This work has been partly funded by the French Agence Nationale de la Recherche through the Decrypt project under Contract ANR-18-CE39-0007. Some of the experiments presented in this article were carried out using the LIMOS' servers. This work has been accepted for presentation at CIFRIS23, the Congress of the Italian association of cryptography "De Componendis Cifris."

Conflict of interest: The authors state that there is no conflict of interest.

Code availability: The code produced for the current study is available in the boomerang_rijndael repository: https://gitlab.com/rloic-gitlab/boomerang-rijndael.

References

[1] Wagner D. The boomerang attack. In: Knudsen LR, editor. FSE'99. vol. 1636 of LNCS. Heidelberg: Springer; 1999. p. 156–70.

[2] Biham E, Shamir A. Differential cryptanalysis of DES-like cryptosystems. In: Menezes AJ, Vanstone SA, editors. CRYPTO'90. vol. 537 of LNCS. Heidelberg: Springer; 1991. p. 2–21.

[3] Biham E, Dunkelman O, Keller N. Related-key boomerang and rectangle attacks. In: Cramer R, editor. Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. vol. 3494 of Lecture Notes in Computer Science. Springer; 2005. p. 507–25. doi: 10.1007/11426639_30.

[4] Cid C, Huang T, Peyrin T, Sasaki Y, Song L. Boomerang connectivity table: a new cryptanalysis tool. In: Nielsen JB, Rijmen V, editors. EUROCRYPT 2018, Part II. vol. 10821 of LNCS. Heidelberg: Springer; 2018. p. 683–714.

[5] Delaune S, Derbez P, Vavrille M. Catching the fastest boomerangs application to SKINNY. IACR Trans Symm Cryptol. 2020;2020(4):104–29.
[6] Daemen J, Rijmen V. AES proposal: Rijndael. 1999.
[7] Derbez P, Euler M, Fouque P, Nguyen PH. Revisiting related-key boomerang attacks on AES using computer-aided tool. In: Agrawal S, Lin D, editors. Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III. vol. 13793 of Lecture Notes in Computer Science. Springer; 2022. p. 68–88. doi: 10.1007/978-3-031-22969-5_3.
[8] Zhou N, Kjellerstrand H. The Picat-SAT compiler. In: Practical Aspects of Declarative Languages - PADL 2016. vol. 9585 of LNCS. Springer; 2016. p. 48–62.
[9] Prud'homme C, Fages JG, Lorca X. Choco documentation; 2016. http://www.choco-solver.org.
[10] Jr JN, Pavaaaao IC. Impossible-differential attacks on large-block Rijndael. In: Garay JA, Lenstra AK, Mambo M, Peralta R, editors. Information security, 10th International Conference, ISC 2007. vol. 4779 of LNCS. Springer; 2007. p. 104–17.
[11] Zhang L, Wu W, Park JH, Koo B, Yeom Y. Improved impossible differential attacks on large-block Rijndael. In: Wu T, Lei C, Rijmen V, Lee D, editors. Information Security, 11th International Conference, ISC 2008. vol. 5222 of LNCS. Springer; 2008. p. 298–315.
[12] Galice S, Minier M. Improving integral attacks against Rijndael-256 Up to 9 rounds. In: Vaudenay S, editor. Progress in Cryptology - AFRICACRYPT 2008. vol. 5023 of LNCS. Springer; 2008. p. 1–15.
[13] Wang Q, Gu D, Rijmen V, Liu Y, Chen J, Bogdanov A. Improved impossible differential attacks on large-block Rijndael. In: Kwon T, Lee M, Kwon D, editors. Information security and cryptology - ICISC 2012. vol. 7839 of LNCS. Springer; 2012. p. 126–40.
[14] Minier M. Improving impossible-differential attacks against Rijndael-160 and Rijndael-224. Des Codes Cryptogr. 2017;82(1–2):117–29. doi: 10.1007/s10623-016-0206-7.
[15] Liu Y, Shi Y, Gu D, Dai B, Zhao F, Li W, et al. Improved impossible differential cryptanalysis of large-block Rijndael. Sci China Inf Sci. 2019;62(3):32101:1–32101:14. doi: 10.1007/s11432-017-9365-4.
[16] Wang Q, Liu Z, Toz D, Varici K, Gu D. Related-key rectangle cryptanalysis of Rijndael-160 and Rijndael-192. IET Inf Secur. 2015;9(5):266–76. doi: 10.1049/iet-ifs.2014.0380.
[17] Daemen J, Rijmen V. The design of Rijndael: AES – the Advanced Encryption Standard. Berlin; London: Springer; 2002. OCLC: 751525895.
[18] Advanced Encryption Standard (AES); 2001. National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce.
[19] Song L, Qin X, Hu L. Boomerang connectivity table revisited. Application to SKINNY and AES. IACR Trans Symmetric Cryptol. 2019;2019(1):118–41. doi: 10.13154/tosc.v2019.i1.118-141.
[20] Jean J. TikZ for Cryptographers; 2016. https://www.iacr.org/authors/tikz/.
[21] Gerault D, Lafourcade P, Minier M, Solnon C. Computing AES related-key differential characteristics with constraint programming. Artif Intell. 2020 Jan;278:103183. https://linkinghub.elsevier.com/retrieve/pii/S0004370218303631.
[22] Rouquette L, Gérault D, Minier M, Solnon C. And Rijndael: automatic related-key differential analysis of Rijndael. In: Batina L, Daemen J, editors. Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa, AFRICACRYPT 2022, Fes, Morocco, July 18–20, 2022, Proceedings. vol. 13503 of LNCS. Springer Nature Switzerland; 2022. p. 150–75.
[23] Nethercote N, Stuckey PJ, Becket R, Brand S, Duck GJ, Tack G. MiniZinc: towards a standard CP modelling language. In: Principles and Practice of Constraint Programming - CP 2007. vol. 4741 of LNCS. Springer; 2007. p. 529–43.
[24] Chu G, Stuckey PJ. Chuffed solver description; 2014. http://www.minizinc.org/challenge2014/description_chuffed.txt.
[25] Canteaut A, Roué J. On the behaviors of affine equivalent Sboxes regarding differential and linear attacks. In: Oswald E, Fischlin M, editors. Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. vol. 9056 of Lecture Notes in Computer Science. Springer; 2015. p. 45–74. doi: 10.1007/978-3-662-46800-5_3.
[26] Daemen J, Rijmen V. Understanding two-round differentials in AES. In: Prisco RD, Yung M, editors. Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6–8, 2006, Proceedings. vol. 4116 of Lecture Notes in Computer Science. Springer; 2006. p. 78–94. doi: 10.1007/11832072_6.
[27] Dong X, Qin L, Sun S, Wang X. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. IACR Cryptol ePrint Arch. 2021;2021:856. https://eprint.iacr.org/2021/856.
[28] 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. 2020;88(6):1103–26. doi: 10.1007/s10623-020-00730-1.
[29] Selçuk AA. On probability of success in linear and differential cryptanalysis. J Cryptol. 2008 Jan;21(1):131–47.
[30] Biere A. Lingeling and friends at the SAT Competition 2011. 2011. Institut for Formal Models and Verification, Johannes Kepler University. https://epub.jku.at/obvulioa/content/titleinfo/5973538.
[31] Libralesso L, Delobel F, Lafourcade P, Solnon C. Automatic generation of declarative models for differential cryptanalysis. In: Michel LD, editor. 27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25–29, 2021. vol. 210 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik; 2021. p. 40:1–40:18. doi: https://doi.org/10.4230/LIPIcs.CP.2021.40.
[32] Biryukov A, Nikolic I. 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. vol. 6110 of LNCS. Springer; 2010. p. 322–44.

Appendix

A Overall probabilities

K_{len}	N_r	D_r	obj_{s1}	obj_{s2}	optimality
128	3	1	2^{0}	2^{-6}	×
128	4	2	2^{0}	2^{-5}	×
128	5	3	2^{0}	2^{0}	✓
128	6	4	2^{-12}	2^{-20}	×
128	7	5	2^{-59}		
160	3	1	2^{0}	2^{-29}	×
160	4	2	2^{0}	2^{0}	✓
160	5	3	2^{0}	2^{0}	✓
160	6	4	2^{0}	2^{0}	✓
160	7	5	2^{-18}	2^{-35}	×
160	8	6	2^{-56}	2^{-104}	×
160	9	7	2^{-84}	2^{-95}	×
192	3	1	2^{0}	2^{0}	✓
192	4	2	2^{0}	2^{-5}	×
192	5	3	2^{0}	2^{0}	✓
192	6	4	2^{0}	2^{0}	✓
192	7	5	2^{0}	2^{0}	✓
192	8	6	2^{-12}	2^{-12}	✓
192	9	7	2^{-47}	2^{-51}	×
192	10	8	2^{-76}	2^{-136}	×
192	11	9	2^{-119}		
224	3	1	2^{0}		
224	4	2	2^{0}		
224	5	3	2^{0}		
224	6	4	2^{0}		
224	7	5	2^{0}		
224	8	6	2^{-24}		
224	9	7	2^{-47}		
224	10	8	2^{-78}		
224	11	9	2^{-90}		
224	12	10	2^{-155}		
256	3	1	2^{0}		
256	4	2	2^{0}		
256	5	3	2^{0}		
256	6	4	2^{0}		
256	7	5	2^{0}		
256	8	6	2^{-12}		
256	9	7	2^{-35}		
256	10	8	2^{-48}		
256	11	9	2^{-64}		
256	12	10	2^{-88}		
256	13	11	2^{-114}		

(a)

K_{len}	N_r	D_r	obj_{s1}	obj_{s2}	optimality
128	3	1	2^{0}	2^{-12}	×
128	4	2	2^{0}	2^{-5}	×
128	5	3	2^{0}	2^{0}	✓
128	6	4	2^{-12}	2^{-18}	✓
160	3	1	2^{0}	2^{-6}	×
160	4	2	2^{0}	2^{0}	✓
160	5	3	2^{0}	2^{0}	✓
160	6	4	2^{-11}	2^{-12}	✓
160	7	5	2^{-41}	2^{-45}	×
160	8	6	2^{-118}		
192	3	1	2^{0}		
192	4	2	2^{0}		
192	5	3	2^{0}		
192	6	4	2^{0}		
192	7	5	2^{0}		
192	8	6	2^{-47}		
192	9	7	2^{-102}		
224	3	1	2^{0}		
224	4	2	2^{0}		
224	5	3	2^{0}		
224	6	4	2^{0}		
224	7	5	2^{0}		
224	8	6	2^{-12}		
224	9	7	2^{-42}		
224	10	8	2^{-114}		
256	3	1	2^{0}		
256	4	2	2^{0}		
256	5	3	2^{0}		
256	6	4	2^{0}		
256	7	5	2^{0}		
256	8	6	2^{-18}		
256	9	7	2^{-36}		
256	10	8	2^{-87}		
256	11	9	2^{-120}		

(b)

K_{len}	N_r	D_r	obj_{s1}	obj_{s2}	optimality
128	3	1	2^{0}	2^{0}	✓
128	4	2	2^{0}	2^{0}	✓
128	5	3	2^{0}	2^{0}	✓
128	6	4	2^{-36}	2^{-36}	✓
160	3	1	2^{0}	2^{-6}	×
160	4	2	2^{0}	2^{0}	✓
160	5	3	2^{0}	2^{0}	✓
160	6	4	2^{-6}	2^{-6}	✓
160	7	5	2^{-46}	2^{-73}	×
192	3	1	2^{0}		
192	4	2	2^{0}		
192	5	3	2^{0}		
192	6	4	2^{0}		
192	7	5	2^{-28}		
192	8	6	2^{-84}		
224	3	1	2^{0}		
224	4	2	2^{0}		
224	5	3	2^{0}		
224	6	4	2^{0}		
224	7	5	2^{-29}		
224	8	6	2^{-54}		
256	3	1	2^{0}		
256	4	2	2^{0}		
256	5	3	2^{0}		
256	6	4	2^{0}		
256	7	5	2^{-6}		
256	8	6	2^{-35}		

(c)

K_{len}	N_r	D_r	obj_{s1}	obj_{s2}	optimality
128	3	1	2^{0}		
128	4	2	2^{0}		
128	5	3	2^{0}		
128	6	4	2^{-24}		
160	3	1	2^{0}		
160	4	2	2^{0}		
160	5	3	2^{0}		
160	6	4	2^{0}		
160	7	5	2^{-54}		
192	3	1	2^{0}		
192	4	2	2^{0}		
192	5	3	2^{0}		
192	6	4	2^{0}		
192	7	5	2^{-36}		

(d)

Figure A1: The probabilities found for the different versions of Rijndael. Each table represents a variant Clen of Rijndael. Nr is the number of rounds, Dr is the number of rounds for which we compute the probability of distinction ($Dr = Nr2$). objs1 is the upper bound found by Step1-Opt. objs2 is the best probability found with Step2-Opt. objs2 is not given either when we did not perform the computation or when the computation did not finish. The optimality column indicates whether the algorithm has found (✓) the optimal bound (complete search) or not (x) (incomplete search).

B Related-key boomerang distinguisher on nine rounds for Rijndeal-128-160

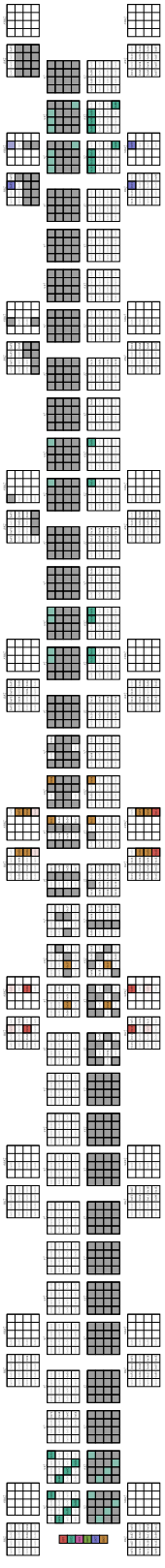

Figure A2: The Rijndael-128-160 nine rounds attack. The distinguisher works for rounds 1–8 and has a probability of 2^{56} for the encryption part and of 2^{39} for the KeySchedule.