On Impossible Boomerang Attacks

Xavier Bonnetain, Margarita Cordero, Virginie Lallemand, Marine Minier, Maria Naya Plasencia

IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2024, No. 2, pp. 222–253. DOI:10.46586/tosc.v2024.i2.222-253

On Impossible Boomerang Attacks

Application to Simon and SKINNYee

Xavier Bonnetain[1], Margarita Cordero[1], Virginie Lallemand[1], Marine Minier[1] and María Naya-Plasencia[2]

[1] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
firstname.name@loria.fr
[2] Inria, Paris, France
maria.naya_plasencia@inria.fr

Abstract. The impossible boomerang attack, introduced in 2008 by Jiqiang Lu, is an extension of the impossible differential attack that relies on a boomerang distinguisher of probability 0 for discarding incorrect key guesses. In Lu's work, the considered impossible boomerang distinguishers were built from 4 (different) probability-1 differentials that lead to 4 differences that do not sum to 0 in the middle, in a miss-in-the-middle way.
In this article, we study the possibility of extending this notion by looking at finer-level contradictions that derive from boomerang switch constraints. We start by discussing the case of quadratic Feistel ciphers and in particular of the Simon ciphers. We exploit their very specific boomerang constraints to enforce a contradiction that creates a new type of impossible boomerang distinguisher that we search with an SMT solver. We next switch to word-oriented ciphers and study how to leverage the Boomerang Connectivity Table contradictions. We apply this idea to SKINNYee, a recent tweakable block cipher proposed at Crypto 2022 and obtain a 21-round distinguisher.
After detailing the process and the complexities of an impossible boomerang attack in the single (twea)key and related (twea)key model, we extend our distinguishers into attacks and present a 23-round impossible boomerang attack on Simon-32/64 (out of 32 rounds) and a 29-round impossible boomerang attack on SKINNYee (out of 56 rounds). To the best of our knowledge our analysis covers two more rounds than the (so far, only) other third-party analysis of SKINNYee that has been published to date.

Keywords: Cryptanalysis · Impossible boomerang attack · Simon · SKINNYee

1 Introduction

Boomerang attacks were introduced in 1999 by David Wagner in [Wag99]. This type of attacks rely on distinguishers exploiting the fact that, for two given differences α and δ, the following relation is verified more often for the block cipher E than for a random permutation:

$$E^{-1}(E(M) \oplus \delta) \oplus E^{-1}(E(M \oplus \alpha) \oplus \delta) = \alpha \tag{1}$$

In this first work, the technique proposed to build such a distinguisher was based on two differentials, one of probability p covering the first rounds (denoted E_0) and the other of probability q covering the next rounds (denoted E_1, where $E = E_1 \circ E_0$). The occurences of these differentials over E_0 and E_1 were considered as independent events that could thus be combined together with probability p^2q^2 to build a distinguisher on E.

Received: 2023-11-23 Revised: 2024-03-01 Accepted: 2024-05-01 Published: 2024-06-18

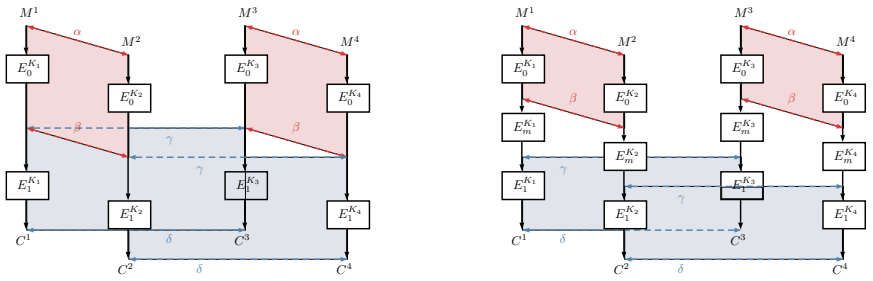

Figure 1: Naive construction of a related-key boomerang distinguisher (left) and sandwich construction (right).

Soon after the introduction of this construction, many publications [Mur11, Kir15, BK09] showed that the underlying strong independence assumption is frequently wrong, resulting in the p^2q^2 probability estimate being far from the actual probability. Researchers then studied how to get a better approximation, starting with the sandwich framework [DKS10] that cuts the cipher in 3 parts instead of 2 ($E = E_1 \circ E_m \circ E_0$) as depicted on the right in Figure 1. The middle part E_m is introduced in order to pick out and study separately the rounds where the top and bottom differential trails intermingle. Following results proposed techniques to study the probability of the middle part in a systematic way, as for instance the BCT framework [CHP+18] for the case where E_m is one round of SPN cipher, and its multiple related works on other cipher types [BHL+20, WWS23] or on more rounds [WP19, SQH19, YSS+22].

In addition to the study of the possible boomerang distinguishers, a perpendicular direction introduced the impossible boomerang distinguisher notion [Lu08]. The idea is similar to what is done for impossible differential distinguishers [Knu98, BBS99a], that is to rely on a pair of differences that never appear together to discard wrong key guesses. The technique devised in [Lu08] relies on 4 different trails leading with probability 1 to 4 (truncated) differences $\beta, \beta', \gamma, \gamma'$ positioned in a middle round and satisfying that $\beta \oplus \beta' \oplus \gamma \oplus \gamma' \neq 0$, which implies the impossibility (see on the left in Figure 2).

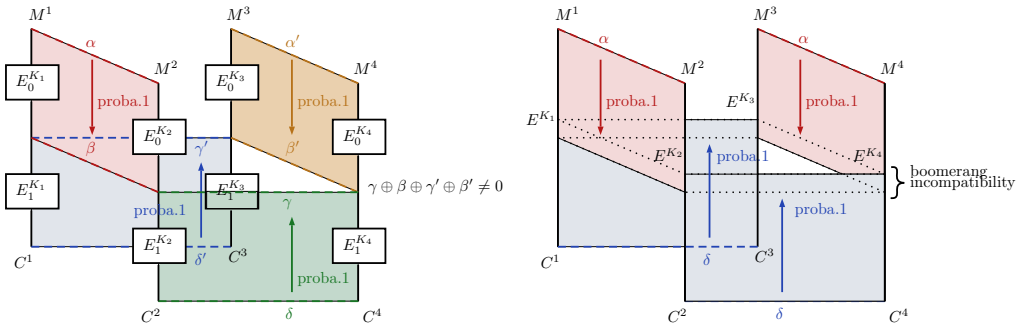

Figure 2: Construction of an impossible boomerang distinguisher as introduced by Lu (left) and a specific case supported by our extension (with $\alpha' = \alpha$ and $\delta' = \delta$) (right).

Our Contributions. In this article we study possible extensions of the theory of [Lu08] by using impossibilities coming from boomerang switch contradictions: for Sbox-based ciphers we show how to leverage the BCT theory and rely on a coefficient of value 0, while we

Table 1: Summary of previous and new results on SIMON-32/64. SK: single-key. RK: related-key scenario and number of related keys. RX: rotational-XOR. FC: full codebook.

Nb Rounds	Type of attacks	Data	Time	Ref.
22	Differential (SK)	2^{32} (FC)	$2^{58.76}$	[QHS15]
23	Linear (SK)	$2^{31.19}$	$2^{56.5}$	[CW16]
24	Integral (SK)	2^{32} (FC)	2^{63}	[CCW+18]
25	RX-Rectangle (**4** RK)	2^{34} (FC)	$2^{59.7}$	[BL23b]
23	Imp. Boom. (**4** RK)	2^{34} (FC)	$2^{61.75}$	Section 5.1.2
23	Imp. Boom. (**16** RK)	2^{36} (FC)	$2^{53.1}$	Section 5.1.3
23	Imp. Boom. (**16** RK)	2^{35}	$2^{59.85}$	Section 5.1.3

Table 2: Summary of previous and new results on SKINNYee. SK: single-key. RT: related-tweak.

Nb Rounds	Type of attacks	Data	Time	Ref.
26	Integral (SK/RT)	2^{66}	2^{113}	[HSE23]
27	Imp. Differential (SK/RT)	$2^{62.79}$	$2^{123.04}$	[HSE23]
29	Imp. Boomerang (SK/RT)	2^{66}	$2^{126.6}$	Section 5.2.2
29	Imp. Boomerang (SK/RT)	$2^{67.2}$	$2^{119.2}$	Section 5.2.3

show how similar ideas can be developed for bit-oriented ciphers and in particular how one can benefit from the simple round function of the ciphers of the SIMON family [BSS+15].

We present distinguishers on the 10 variants of SIMON and extend the more promising one into a 23-round attack on SIMON-32/64 (see Table 1), reaching two less rounds than the current state of the art [BL23b] (that was obtained with a standard boomerang technique). We apply our technique based on the BCT contradiction to the recent tweakable block cipher SKINNYee [NSS22] and propose a 29-round attack, beating by 3 and 2 rounds the integral and impossible differential attacks proposed in the (up to our knowledge) only existing third-party analysis article [HSE23].

Outline. After recalling some definitions, we give in Section 2 a short review of the boomerang incompatibilities reported in the literature, that we discuss in Section 3 from the perspective of building impossible boomerang distinguishers. We next propose two extensions of the impossible boomerang distinguisher of Lu and provide in Section 4 a thorough analysis of the complexity of an impossible boomerang attack, detailing the single (twea)key case and the related (twea)key scenario, for two possible approaches. Last but not least, we present two applications of our new technique in Section 5: one 23-round attack on SIMON-32/64 (and distinguishers of the other versions) and a 29-round attack on SKINNYee (see Table 2).

2 Preliminaries

2.1 Boomerang Tables

In this subsection we recall the classical definitions of the boomerang tables, namely the BCT, the UBCT and the LBCT, that were introduced to easily compute the probability

of 1-round boomerang switches for SPN ciphers. We also give the definition of the Feistel variant, the FBCT.

Definition 1 (BCT [CHP+18])**.** Let S be a permutation of $\mathbb{F}_2^n$, and $\alpha_1, \beta_2 \in \mathbb{F}_2^n$. The Boomerang Connectivity Table (BCT) of S is a two-dimensional table defined by:

$$BCT(\alpha_1, \beta_2) = \#\{x \in \mathbb{F}_2^n \mid S^{-1}(S(x) \oplus \beta_2) \oplus S^{-1}(S(x \oplus \alpha_1) \oplus \beta_2) = \alpha_1\}.$$

Definition 2 (UBCT [WP19, DDV20])**.** Let S be a permutation of $\mathbb{F}_2^n$. The Upper Boomerang Connectivity Table (UBCT) of S is a three-dimensional table defined by:

$$\begin{aligned}\mathrm{UBCT}(\alpha_1, \alpha_2, \beta_2) = \#\{x \in \mathbb{F}_2^n \mid S^{-1}(S(x) \oplus \beta_2) \oplus S^{-1}(S(x \oplus \alpha_1) \oplus \beta_2) = \alpha_1,\\ S(x) \oplus S(x \oplus \alpha_1) = \alpha_2\} \text{ where } \alpha_1, \alpha_2, \beta_2 \in \mathbb{F}_2^n.\end{aligned}$$

Definition 3 (LBCT [WP19, DDV20])**.** Let S be a permutation of $\mathbb{F}_2^n$. The Lower Boomerang Connectivity Table (LBCT) of S is a three-dimensional table defined by:

$$\begin{aligned}\mathrm{LBCT}(\alpha_1, \beta_1, \beta_2) = \#\{x \in \mathbb{F}_2^n \mid S^{-1}(S(x) \oplus \beta_2) \oplus S^{-1}(S(x \oplus \alpha_1) \oplus \beta_2) = \alpha_1,\\ S(x) \oplus S(x \oplus \beta_1) = \beta_2\} \text{ where } \alpha_1, \beta_2, \beta_1 \in \mathbb{F}_2^n.\end{aligned}$$

Similar tables were introduced to deal with Feistel ciphers using Sboxes in their round function. The counterpart of the BCT is the FBCT, defined as follows:

Definition 4 (FBCT [BHL+20])**.** Let S be a function from $\mathbb{F}_2^n$ to itself. The Feistel Boomerang Connectivity Table (FBCT) is a two-dimensional table defined by:

$$FBCT(\alpha_1, \beta_2) = \#\left\{x \in \mathbb{F}_2^n \,|S(x) \oplus S(x \oplus \alpha_1) \oplus S(x \oplus \beta_2) \oplus S(x \oplus \alpha_1 \oplus \beta_2) = 0\right\}.$$

2.2 Previous Impossible Boomerang Notions

Since the advent of boomerang distinguishers, many works have remarked the actual probability of the distinguisher can be vastly different than predicted by a model. In particular, in some cases it was argued the probability of the distinguisher was actually 0, that is, it is impossible. However, there are two very different notions of impossibility:

Incompatible characteristics. The probability of a boomerang distinguisher can be estimated from the probability of some characteristics, using some approximation techniques. It might be that these characteristics are incompatible, that is, it is not possible to have a quartet of messages that follows the characteristics considered.

Impossible boomerang distinguisher. There exist no messages M that satisfy the boomerang equation (Equation (1)).

The first notion allows to claim an attack is invalid, as the argument supporting the existence of a high-probability boomerang distinguisher does not hold. It is however weaker than the second notion, as by random chance the boomerang might still occur, using unaccounted for characteristics.

We detail in what follows different examples from the literature.

2.2.1 Results by Murphy

The initial probability estimation techniques for boomerang probabilities were criticized by Murphy in [Mur11]. He proposed boomerang distinguishers constructed from characteristics on DES and AES that were predicted to have a given probability, but actually have probability 0, or, as Murphy wrote:

"This [DES/AES] boomerang *never* comes back."

These examples are only incompatible characteristics, as part of the differential propagation in the cipher is fixed.

2.2.2 Results by Dunkelman, Keller and Shamir

In the journal version of the article introducing the sandwich framework [DKS14], Dunkelman and co-authors presented two boomerang characteristics of probability zero: one obtained when slightly modifying the key schedule of the considered cipher, and the second when slightly changing the distinguisher. The stress was put on the importance of thoroughly estimating the probability of the connecting part of E_m.

2.2.3 The Impossible Boomerang Attack

The so-called impossible boomerang attack is a cryptanalysis technique introduced by Jiqiang Lu. He first introduced it in his PhD Thesis [Lu08] and published it later in the article [Lu11]. Described as a combination of the boomerang attack and of the impossible differential cryptanalysis, it relies on an impossible boomerang distinguisher that, as its name indicates, is a boomerang distinguisher of probability 0. To build such a distinguisher, the author starts by rewriting the cipher E as the composition of two sub-ciphers ($E = E_1 \circ E_0$) and searches for 4 (truncated) probability-1 differentials: two over E_0, and two over E_1^{-1}, with the condition that the 4 differences in the middle do not sum to zero. Following the notation on the left in Figure 2, these are denoted $\alpha \xrightarrow{E_0} \beta$, $\alpha' \xrightarrow{E_0} \beta'$, $\delta \xrightarrow{E_1^{-1}} \gamma$ and $\delta' \xrightarrow{E_1^{-1}} \gamma'$, respectively, with $\beta \oplus \beta' \oplus \gamma \oplus \gamma' \neq 0$.

The middle contradiction implies that it is impossible to observe together that $E(M_1) \oplus E(M_3) = \delta'$ and $E(M_1 \oplus \alpha) \oplus E(M_3 \oplus \alpha') = \delta$, and this observation can be turned into a key recovery attack. Similarly to the impossible-differential case, it consists in adding a few rounds before and/or after the distinguisher, making (partial) round key guesses to compute the differences at the start and at the end of the distinguisher, and discarding any key guess that would imply the previous impossible relations.

As explained by Lu, the single-key impossible boomerang distinguisher is actually of little interest, as it directly implies that an impossible differential distinguisher of the same size can be obtained (and, a priori, it can be turned into a more efficient attack as it is based on pairs of messages instead of quartets). We recall below the proposition 1 from [Lu11]:

Proposition 1 (Limitation of the single-key impossible boomerang attack [Lu11])**.** *From a single key impossible boomerang distinguisher, an impossible differential for the same number of rounds can be obtained. A block cipher resistant to related-key impossible differential cryptanalysis will not necessarily resist a related-key impossible boomerang attack.*

The proof of the first assertion uses the fact that the contradiction $\beta \oplus \beta' \oplus \gamma \oplus \gamma' \neq 0$ (as can be seen on the left in Figure 2) can be rewritten as $\beta \oplus \gamma \neq \beta' \oplus \gamma'$ which implies that at least either $\beta \oplus \gamma \neq 0$ or $\beta' \oplus \gamma' \neq 0$. It leads to an impossible differential distinguisher over $E = E_1 \circ E_0$ based on two probability-1 differentials, one over E_0 with an input difference of α (resp. α') and the other over E_1^{-1} starting with a difference of δ (resp. δ') that contradict in the middle.

The related-key impossible boomerang attack is the straightforward extension of the impossible boomerang attack to the related-key case, where keys are managed as in the related-key boomerang attack [BDK05, HKLP05]. This scenario might have advantages over the impossible differential one, mainly because two key differences can be used. This technique was applied to AES in [Lu11].

2.2.4 Results by Peyrin and Tan

In [PT22], the authors present characteristics that are possible assuming round independence but that are actually either key-dependent (meaning that the characteristic is

actually a weak-key characteristic), or even actually impossible. Both differential and boomerang distinguishers based on such characteristics are identified in published papers. This is another example of incompatible characteristics, this time from the fact that one differential characteristic has probability 0.

2.2.5 The Double Boomerang Connectivity Table

Finally, the Double Boomerang Connectivity Table is a two-dimensional table named in [HBS21] and studied in detail in [YSS+22]. Its goal is to capture the probability that a boomerang comes back over two consecutive Sboxes. Its definition relies on the random subkey assumption. It is similar to the BCT, but over two rounds, as shown in Figure 3. We study this table in more detail in Section 3.1.

Definition 5 (DBCT [HBS21, YSS+22])**.** The Double Boomerang Connectivity Table (DBCT) of an Sbox S is defined from its UBCT and LBCT as:

$$DBCT(\alpha_1, \beta_3) = \sum_{\alpha_2, \beta_2} UBCT(\alpha_1, \alpha_2, \beta_2) \cdot LBCT(\alpha_2, \beta_2, \beta_3).$$

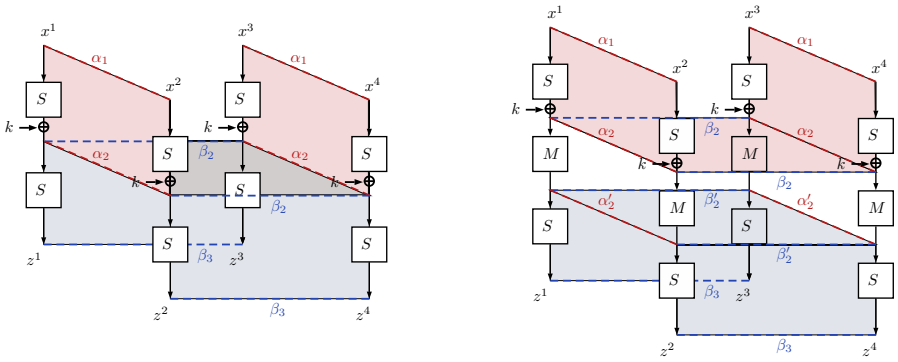

Figure 3: Parameters of a DBCT (left) and of a general DBCT with a linear layer in the middle (right). A UBCT is computed at the top (in red) and an LBCT is computed at the bottom (in blue). Notations from [YSS+22].

In [YSS+22], a general DBCT is used to claim some 2-round transitions in some boomerang characteristics from [CDJ+20, BL23a] have probability 0, which invalidates the corresponding distinguishers.

3 New Insights on Boomerang Impossibilities

3.1 The DBCT Theory

We focus here on the general setting represented on the right in Figure 3 that takes into account the linear layer between Sboxes, corresponding to the following definition:

Definition 6 (DBCT (general case) [YSS+22])**.** Let S be an Sbox layer, M a linear layer. The Double Boomerang Connectivity Table (DBCT) of the cipher $S \circ M \circ S$ is defined as:

$$DBCT(\alpha_1, \beta_3) = \sum_{\substack{\alpha'_2 = M(\alpha_2) \\ \beta'_2 = M(\beta_2)}} UBCT(\alpha_1, \alpha_2, \beta_2) \cdot LBCT(\alpha'_2, \beta'_2, \beta_3).$$

Using this variant of the DBCT, it was claimed that boomerang distinguishers from [CDJ+20, BL23a] were invalid.

Limits of the DBCT. The DBCT only considers transitions where, in the middle, the quartet of differences has the form[1] (before and after the linear layer) $(\overline{\alpha}, \overline{\beta}, \overline{\alpha}, \overline{\beta})$ where $\overline{\alpha}$ and $\overline{\beta}$ are some differences. This is however a restriction as it misses the admissible set of middle differences $(\alpha, \beta, \gamma, \alpha \oplus \beta \oplus \gamma)$ with $\alpha \neq \gamma$. Hence, having $DBCT(\alpha_1, \beta_3) = 0$ does not imply that the two-round boomerang distinguisher with input difference α_1 and output difference β_3 has probability 0. Thus, the analysis of [YSS+22] is partial, as the $DBCT$ by itself cannot prove a 2-round boomerang is impossible.

On the possibility of general differences. We now study in which cases it is impossible to have middle differences not of the form $(\overline{\alpha}, \overline{\beta}, \overline{\alpha}, \overline{\beta})$.

Theorem 1 (DBCT with restricted activity pattern.)**.** *Let α_1, β_3 be two differences. If there are no non-zero differences α', β' such that α' has a sparser activity pattern than α_1, β' a sparser activity pattern than β_3 and $\beta' = M\alpha'$, then the middle differences in the two-round boomerang distinguisher defined by α_1 and β_3 must be of the form $(\overline{\alpha}, \overline{\beta}, \overline{\alpha}, \overline{\beta})$. In this case, the DBCT actually conveys the probability, on average over the middle round keys, of the corresponding 2-round boomerang distinguisher.*

Remark 1. In the above theorem, "a sparser activity pattern" has to be understood as "inactive Sboxes are still inactive, active Sboxes can be active".

Proof. We note α_2^L, α_2^R the differences from α_1 after the first Sbox layer (on the left and on the right sides, respectively), and $\beta_2'^F, \beta_2'^B$ the differences before the second Sbox layer that lead to β_3 (on the front and back sides, respectively). As the sum of the four differences must be 0, after the linear layer we must have:

$$M(\alpha_2^L) \oplus \beta_2'^F \oplus M(\alpha_2^R) \oplus \beta_2'^B = 0 \Leftrightarrow M(\alpha_2^L \oplus \alpha_2^R) = \beta_2'^F \oplus \beta_2'^B.$$

α_2^L and α_2^R (resp. $\beta_2'^F$ and $\beta_2'^B$) have the same activity pattern as α (resp. β). Hence, if there is no non-zero differences $\overline{\alpha}$, $\overline{\beta}$ with a sparser activity pattern than α and β such that $M(\overline{\alpha}) = \overline{\beta}$, then we must have $\alpha_2^L = \alpha_2^R$ and $\beta_2'^F = \beta_2'^B$.

Finally, as the differences not considered by the DBCT cannot appear in the middle of the boomerang configuration in this case, the DBCT gives the number of solutions of the boomerang equation. □

Corollary 1. *In the case of AES, all 2-round boomerang distinguishers whose differences have overall at most 4 active Sboxes per super-Sbox are captured by the DBCT.*

DBCT with related-tweak/related-keys. With key or tweak differences, the differences in the middle are a bit more complex. If the tweakey addition is before the linear layer, then the constraint becomes:

$$M(\alpha_2^L \oplus \Delta T_1) \oplus \beta_2'^F \oplus M(\alpha_2^R \oplus \Delta T_2) \oplus \beta_2'^B = 0 \Leftrightarrow M(\alpha_2^L \oplus \alpha_2^R \oplus \Delta T_1 \oplus \Delta T_2) = \beta_2'^F \oplus \beta_2'^B.$$

In general, the tweakey difference is the same for one pair of messages and for the other pair of messages in the quartet. In this case, the constraint is not changed. Otherwise, instead of considering the activity pattern of α, we have to consider the activity pattern of $(\alpha \oplus \Delta T_1 \oplus \Delta T_2)$, that is, the input and the difference of tweak differences. Of course, if the tweakey addition is after the linear layer, the situation is reversed and one has to consider, instead of β, the activity pattern of $(\beta \oplus \Delta T_1 \oplus \Delta T_2)$.

[1] in Figure 3 it corresponds to the differences $(\alpha_2, \beta_2, \alpha_2, \beta_2)$ before M and to $(\alpha_2', \beta_2', \alpha_2', \beta_2')$ after it.

Impact on the claims. With the above theorem, it is possible to confirm the 2-round impossible boomerang configurations detected in [YSS+22], as the studied characteristics are on AES-based ciphers and have differences with a sparse enough activity pattern (with super-Sboxes containing either 2 or 3 active Sboxes).

Generalization. In the recent article [WSW+23], a variant of the DBCT that allows all possible differences, called the DBCT*, is proposed. This tool is able to detect any key-independent 2-round impossible boomerang distinguisher. Note that the impact on the previous work is not investigated in [WSW+23].

3.2 Extending the Impossible Boomerang Distinguisher Notion by using New Contradictions

Our idea is to study the possibility to define an impossible boomerang distinguisher using the miss-in-the-middle principle [BBS99b] where the *miss* event corresponds to a boomerang incompatibility instead of a simple inequality of the differences as done in [Lu11].

The parameters used in this extension are the same as previously: a set of 4 internal state differences α, α', δ, δ' together with two key differences[2] defining four keys (see on the right in Figure 2). The top differences (α, α') are propagated with probability 1 in the encryption direction, and the bottom differences (δ, δ') are propagated with probability 1 in the decryption direction, under the corresponding key differences. This set of parameters leads to an impossible boomerang distinguisher if a *boomerang incompatibility* occurs at some step.

Contradicting Boomerang Switches. Our main idea to build new impossible boomerang distinguishers is to propagate differences with probability 1 so that an impossible boomerang switch (over one or two rounds) appears in a deterministic way in the middle. This impossible boomerang switch can be expressed in the form of boomerang tables contradictions, or in an ad hoc manner. A first example could be to obtain a one-round contradiction corresponding to a null BCT coefficient (for an SPN cipher) or to a null FBCT coefficient (for a Feistel cipher using Sboxes). The GBCT and FGBCT [LWL22] (which are generalization of these tables in which differences on the facing sides are not equal) could also be used. In the case of a two-round contradiction and as discussed above in Section 3.1, the correct tool to catch a contradiction is the DBCT*. Again, one could also consider an extended version where the differences are not equal on facing sides.

We remark here that with Lu's technique, the attacker must set $\alpha \neq \alpha'$ or $\delta \neq \delta'$ as these differences are propagated with probability one and must lead to four differences that do not sum to 0 in the middle (see Figure 2). When considering contradictions based on one or two-round impossible boomerang switches, this restriction does not hold anymore and an attacker might also select $\alpha = \alpha'$ and $\delta = \delta'$. Our new framework thus extends the framework of [Lu11] (as an inequality implies a boomerang incompatibility, the previous distinguishers are included) and allows a larger set of possibilities for the initial differences.

In our attacks on Simon and SKINNYee (discussed in Section 5), we focus on the case where $\alpha = \alpha'$ and $\delta = \delta'$. If this specific case looks restrictive at first sight, it actually simplifies both the search with the automatic tool (as only two differential trails need to be build) and the key recovery procedure (as the same partial decryptions can be made on facing sides of the boomerang).

Boomerang incompatibilities for Sbox-based ciphers. The first extension of Lu's framework we propose is to look at boomerang contradictions that happen through one round of

[2]The same technique can of course be adapted to related tweak or related tweakey scenarios.

an Sbox-based cipher. Instead of looking for the inequality $\beta \oplus \beta' \oplus \gamma \oplus \gamma' \neq 0$ connecting differences at the same level, we look for a pair of differences $(a, b) \in (\mathbb{F}_2^n)^2$ positioned before and after one round R, in the top and bottom trail respectively so that:

$$\forall X \in \mathbb{F}_2^n \qquad R^{-1}(R(X) \oplus b) \oplus R^{-1}(R(X \oplus a) \oplus b) \neq a.$$

For Sbox-based ciphers this relation has been studied in detail (in the context of possible distinguishers), and the study of the probability of a 1-round boomerang configuration was actually made systematic by the use of a table (that can be seen as the counterpart of the Difference Distribution Table but for boomerang relations instead of differential relations) named the BCT (Boomerang Connectivity Table) for SPN ciphers [CHP+18] and the FBCT (Feistel Boomerang Connectivity Table) for Feistel ciphers [BHL+20].

In the impossible boomerang context a similar approach can be used. The fact that the Sboxes are applied in parallel on different parts of the state allows to rewrite the previous relation for one round as relations for one Sbox each. An attacker can now focus on the reduced problem of finding probability-1 differential trails that lead to differences a_i and b_i (in $\mathbb{F}_2^s$) corresponding to the boomerang table of one Sbox S of s bits, in the top and bottom trail respectively, so that:

$$\begin{cases} BCT(a_i, b_i) = 0 & \text{for an SPN cipher,} \\ FBCT(a_i, b_i) = 0 & \text{for a Feistel cipher.} \end{cases} \tag{2}$$

We detail an application of this idea to the SKINNYee cipher in Section 5.2.

Boomerang incompatibility for quadratic Feistel ciphers. The idea of looking for boomerang incompatibilities over 1 round can be extended to bit-oriented ciphers, with the difficulty that the attacker must know how to guarantee an incompatibility. We develop this idea for quadratic Feistel ciphers as it was observed in [BL23b] that this type of ciphers fall into the extreme case where a one-round boomerang can only have a probability of 0 or 1, depending on the differences at play (but not on the state value).

3.2.1 Bit-Oriented Ciphers: The Quadratic Feistel Ciphers Case

We consider a boomerang construction made from 4 differences that are propagated with probability 1 through the middle of the cipher, and we study how to guarantee that the distinguisher is of probability 0. After only a few rounds of probability-1 propagation, some bit differences become undetermined, and there is no guarantee that a given unknown bit difference on a side is equal to the corresponding bit difference on the facing side of the boomerang distinguisher (i.e. in the other pair in the quartet), even if the same difference has been considered at the beginning (i.e., even if $\alpha = \alpha'$). This observation means in particular that we need to be careful when defining the boomerang incompatibility, as two sides might not follow the same differential characteristic.

We thus consider the generic case in Figure 4 where possibly all differences are distinct and try to determine how to guarantee a contradiction. We denote by f the Feistel cipher's round function and write the equalities that must be verified for a boomerang made of 4 different differences to come back:

$$\begin{cases} f(x_\ell) \oplus f(x_\ell \oplus \delta'_r) \oplus f(x_\ell \oplus \alpha_\ell) \oplus f(x_\ell \oplus \alpha_\ell \oplus \delta_r) = \alpha_r \oplus \alpha'_r \oplus \delta_\ell \oplus \delta'_\ell, \\ \delta_r \oplus \delta'_r \oplus \alpha_\ell \oplus \alpha'_\ell = 0. \end{cases} \tag{3}$$

To build an impossible boomerang distinguisher in a miss-in-the-middle way we need to come up with 4 starting differences that lead with probability 1 to a contradiction of one of these two equations for at least one round.

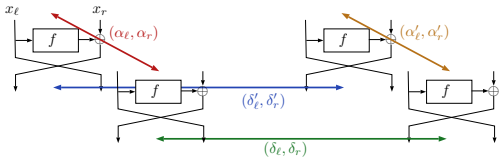

Figure 4: Generic boomerang configuration over one round of a Feistel cipher.

The difficulty of finding such a contradiction depends on the details of the cipher. Inspired by what was presented in [BL23b], we study the case of the quadratic Feistel cipher Simon. Equation (3) applied to Simon gives:

$$\begin{cases} (\delta'_r \lll 2) \oplus (\delta_r \lll 2) \oplus (x_\ell \lll 1)((\delta'_r \oplus \delta_r) \lll 8) \oplus ((\delta'_r \oplus \delta_r) \lll 1)(x_\ell \lll 8) \\ \oplus(\delta'_r \lll 1)(\delta'_r \lll 8) \oplus (\delta_r \lll 1)(\delta_r \lll 8) \oplus (\delta_r \lll 1)(\alpha_\ell \lll 8) \oplus (\alpha_\ell \lll 1)(\delta_r \lll 8) \\ \qquad \oplus\alpha_r \oplus \alpha'_r \oplus \delta_\ell \oplus \delta'_\ell = 0 \\ \qquad \delta_r \oplus \delta'_r \oplus \alpha_\ell \oplus \alpha'_\ell = 0 \end{cases} \tag{4}$$

As we want to be able to conclude without knowing the value of the internal state bits (x_ℓ), we fix the additional constraint that $\delta'_r = \delta_r$ (which implies that $\alpha'_\ell = \alpha_\ell$). The previous relations are reduced to:

$$\begin{cases} (\delta_r \lll 1)(\alpha_\ell \lll 8) \oplus (\alpha_\ell \lll 1)(\delta_r \lll 8) \\ \qquad \oplus\alpha_r \oplus \alpha'_r \oplus \delta_\ell \oplus \delta'_\ell = 0 \\ \qquad \delta_r \oplus \delta'_r = 0 \\ \qquad \alpha_\ell \oplus \alpha'_\ell = 0 \end{cases} \tag{5}$$

In conclusion, a possibility to find an impossible boomerang distinguisher for these ciphers would be to ensure that the last two equalities are verified (so that only the differences are required to conclude) while the first one is not.

3.2.2 Sbox-Based Ciphers: Using BCT Contradictions

The idea is to find α and δ so that when propagated with probability 1 these differences give a BCT configuration that is of probability 0. In practice, it means that we want that in a given round the difference entering an Sbox of the top trail is known and equal to a non-zero value t while in the bottom trail the difference corresponding to the output of the same Sbox of the same round is also known and is equal to $v \neq 0$ so that $BCT(t, v) = 0$.

3.2.3 Sbox-Based Ciphers: Using DBCT Contradictions

A natural extension of distinguishers based on BCT contradictions are distinguishers based on DBCT/DBCT* contradictions. This technique appears advantageous at first, as it would allow to capture contradictions not seen by the BCT and that directly cover 2 rounds. However, it requires to build the DBCT* (or at least some of its coefficients), which is a computationally expensive process, and it also requires to build a distinguisher in which both the input and output differences of a super-Sbox are known (while for ensuring a BCT contradiction only a known input and output Sbox differences are required).

In practice, we observed that having a known difference for a full super-Sbox was a very strong constraint, and it is likely that distinguishers relying on a BCT contradiction can cover more rounds and provide overall a longer distinguisher than those based on DBCT* contradictions.

4 Complexity Analysis of an Impossible Boomerang Attack

In this section we will discuss and provide the complexities of impossible boomerang attacks, considering the distinguishers and the added key recovery rounds. As it was originally done in [BNS14] for impossible differential attacks, we provide some lower-bound formulas on the time and memory complexities, that are matched assuming the early abort guess-and-filter step can be done efficiently. In all cases, a careful check should be done for concrete applications.

We will start by determining the number of needed quartets and their associated data complexity regarding a generic attack configuration for a block cipher in the single key scenario. We will then discuss two methods to conduct the full attack and detail their complexities. Next, we will analyze the related tweakeys case, and finally we will discuss the multiple tweakeys setting and its effect on the complexity.

4.1 Data Complexity

As depicted in Figure 5, we consider an attack based on an impossible boomerang distinguisher E_d extended by some rounds at the top and at the bottom for the key recovery. n is the block size, k is the master key size and we denote by Δ_{in} and Δ_{out} the set of input and output differences that may lead to the distinguisher ends α and β. The size of these sets are denoted $|\Delta_{in}| = 2^{d_{in}}$ and $|\Delta_{out}| = 2^{d_{out}}$. We let $2^{-c_{in}}$ and $2^{-c_{out}}$ denote the probability of reaching the distinguisher differences α and β from differences in Δ_{in} and Δ_{out}. K_{in} and K_{out} are the key bits involved in the differential propagation before and after the distinguisher, respectively.

Note that if the distinguisher only allows for a unique (i.e. fully specified, not truncated) input (resp., output) difference, then $c_{in} = d_{in}$ (resp. $c_{out} = d_{out}$).

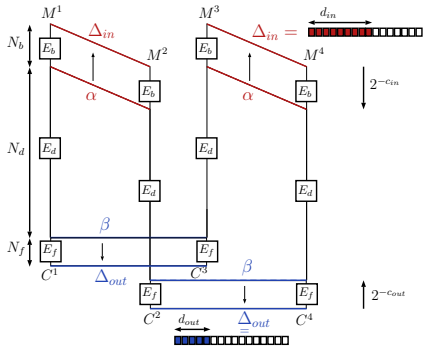

Figure 5: Parameters and setup for a key sieving phase with an impossible boomerang distinguisher E_d. Depending on the considered distinguisher, α and β might be fully specified differences or a set of differences.

Without loss of generality, we assume the queries are performed to the encryption oracle, but we could also choose to perform them to the decryption oracle.

The amount of data needed is given by $\mathcal{D} = 2^s \times 2^{d_{in}}$, which corresponds to the encryption of 2^s structures of $2^{d_{in}}$ messages each, where in a structure the difference between any two elements is in the set Δ_{in}.

From such an amount of data, it is possible to construct $\mathcal{P} = 2^s \times 2^{2d_{in}-1}$ pairs, and from these pairs $\mathcal{Q} = 2^{2s} \times 2^{4d_{in}-2} \times 2^{-2n+2d_{out}}$ quartets that have differences in Δ_{in} and Δ_{out}[3]. As the probability for a given quartet to discard a key is $2^{-2c_{in}-2c_{out}}$, the

[3]While the quartets, as the pairs, are not ordered, from two pairs (x, x') and (y, y'), we can make two distinct quartets, (x, x', y, y') and (x, x', y', y).

probability of not discarding a key is:

$$(1 - 2^{-2c_{in}-2c_{out}})^{\mathcal{Q}} = p,$$

and we have the following approximation:

$$(1 - 2^{-2c_{in}-2c_{out}})^{\mathcal{Q}} \simeq \exp\left(-\mathcal{Q} \times 2^{-2c_{in}-2c_{out}}\right).$$

Overall, the fraction of keys to test is:

$$p \simeq \exp\left(-2^{2s+4d_{in}+2d_{out}-2n-2-2c_{in}-2c_{out}}\right)$$

Extreme case. If we consider a block cipher in a single-key scenario with an impossible boomerang distinguisher with fixed input and output differences, we cannot query more than the full codebook. At best the percentage of keys that remain to be tested is $p \simeq \exp\left(-2^{-2}\right) \simeq 78\%$. This is slightly worse than with impossible differentials, where using the full codebook we only need to test $\exp\left(-2^{-1}\right) \simeq 60\%$ of the keys.

Related-key/Related tweaks. When the encryption function is not the same for all inputs in the quartet, the equations are essentially the same, except that we need to do distinct queries for each function, and that we no longer lose a factor 2 when doing pairs and quartets, that is, $\mathcal{D} = 4 \times 2^{s+d_{in}}$, $\mathcal{P} = 2 \times 2^{s+2d_{in}}$, $\mathcal{Q} = 2^{2s+4d_{in}+2d_{out}-2n}$.

Up to some small factors, this corresponds to the single-key case. We present below two distinct approaches for key recovery.

4.2 Key Recovery with Quartet Filtering

In the first technique we start by generating the data, we consider all the quartets satisfying Δ_{in} and Δ_{out} as in the boomerang configuration, and next, for each key guess, we determine the quartets leading to the impossible differences of the distinguisher (α and β) and discard the guess if some quartets survive. In this case we have:

- Cost of data generation: $\mathcal{D}$
- Cost of producing $\mathcal{Q}$: $\mathcal{P} + \mathcal{Q}$
- Cost of early abort guess-and-filter: $C_{guess} \times \mathcal{Q} \times \frac{2^{|K_{in} \cup K_{out}|}}{2^{2c_{in}+2c_{out}}}$
- Cost of final exhaustive search: $p2^k = \exp\left(-\mathcal{Q} \times 2^{-2c_{in}-2c_{out}}\right) 2^k$.

In the above formula, we assume managing the list of pairs and producing quartets has a cost of 1 encryption per element. Moreover, C_{guess} is the cost of the partial encryption and decryption (with early abort) made in order to verify if a given quartet is compatible with the current key guess. We set its cost to be equal to 1 round of encryption.

The term $-2c_{in} - 2c_{out}$ corresponds to the probability of reaching the impossible boomerang differences (α and β) from the quartet, and can be adapted to cases where not all input and output differences lead to an impossibility.

The memory complexity will be determined by the cost of storing the data and the pairs, as we need to find collisions between pairs to construct the quartets. We do not need to store the discarded keys as we can check keys on-the-fly if no quartet remains after the guess-and-filter procedure:

$$\mathcal{M} = \mathcal{D} + \mathcal{P} + \mathcal{Q}.$$

4.3 Key Recovery with Pair Filtering

The second method resemble (to some extend) the traditional key recovery step in boomerang attacks, as described for instance in [ZDM+20]. Instead of computing all the needed potential quartets for the attack that satisfy Δ_{in} and Δ_{out}, as in the previous step, we use the data differently.

Before building the quartets, we guess the key bits K_{in} involved in the upper part (or K_{out} if it is better, without loss of generality) and with it we build the pairs (M^1, M^2) and (M^3, M^4) that satisfy one of the input differentials at the beginning of the impossible boomerang distinguisher. We next produce quartets from these pairs by selecting (M^1, M^2) and (M^3, M^4) so that $C^1 \oplus C^3$ and $C^2 \oplus C^4$ are part of the Δ_{out} set of differences. This procedure is described in Algorithm 1.

We detail the cost when there is only one input differential α to the distinguisher, and when the related-key setting is considered. The formulas can easily be adapted to the general case, but the details will depend on the constraints of the distinguisher (for example, shall the input differences be equal for the two pairs of a quartet?).

Algorithm 1 Key Recovery of a Related-Key Impossible Boomerang Attack.

1: Construct 2^s structures of size $2^{d_{in}}$, encrypting messages under the 4 keys.
2: **for all** possible values for the top key bits **do**
3: From the structures and the key guess, build (M^1, M^2) and (M^3, M^4) satisfying
4: $M^2 = E_b^{-1}(E_b(M^1) \oplus \alpha)$ and $M^4 = E_b^{-1}(E_b(M^3) \oplus \alpha)$
5: Search for collisions on the positions of the $n - d_{out}$ expected inactive bits of the ciphertexts of M^1 and M^3 and of M^2 and M^4
6: **for all** possible values for the bottom key bits **do**
7: Test the bottom key bits with an early abort process
8: **end for**
9: If no quartet remains, check if the key guess is correct by brute-forcing the remaining bits.
10: **end for**

In this case, the complexities are the following:

- Cost of data generation: $\mathcal{D} = 4 \times 2^{s+d_{in}}$
- Cost of building pairs leading to α: $2^{|K_{in}|} \times 2 \times 2^{s+d_{in}} \times \frac{2|E_b|}{|E|}$
- Cost of producing quartets from the pairs:

$$2^{|K_{in}|} \times 2^{2s+2d_{in}+2d_{out}-2n}$$

- Cost of early abort guess-and-filter:

$$2^{|K_{in}|} \times C_{guess} \times 2^{2s+2d_{in}+2d_{out}-2n} \times \frac{2^{|K_{out} \setminus K_{in}|}}{2^{2c_{out}}}$$

- Cost of final exhaustive search: $p2^k = \exp\left(-2^{2s+2d_{in}+2d_{out}-2n-2c_{out}}\right) 2^k$

In the second step $|E_b|$ is the number of rounds before the distinguisher and $|E|$ is the total number of attacked rounds. As we do a partial encryption then a decryption, we process $2|E_b|$ rounds at this step.

The data complexity stays the same. The memory complexity will be determined by the cost of storing the data and the quartets.

4.4 Discussion on the Possible Key Recovery Approaches

The main specificity of the second approach is the elaborate way of building pairs: at the expense of guessing the required input key bits, the attacker is assured that the pairs fulfill the input difference of the distinguisher. If we compare the two formulas, the only difference is that in the second approach, there are $2^{2d_{in}}$ times less quartets, at the expense of computing the quartets $2^{|K_{in}|}$ times. Hence, as long as $|K_{in}| < 2d_{in}$, the second approach is better.

One advantage of the first technique is that the attacker can more freely organize the early abort phase, possibly by alternating partial encryptions in the upper part with partial decryption in the lower part (to reach strong filters first), while the second approach first fully processes the upper part E_b, and then does partial decryptions on E_f.

Several attacks on AES are presented in the publications introducing the notion of impossible boomerang attacks [Lu08, Lu11]. The related-key impossible boomerang attack on 8-round AES-192 presented in [Lu11] uses a technique very similar to the technique we detail in Section 4.2, while the 9-round related-key attack on AES-256 of [Lu11] builds pairs like in Section 4.3. The author however uses an ad hoc analysis, and to the best of our knowledge we are the first to provide generic procedures with formulas. Interestingly, from our understanding, Lu uses a probabilistic extension in the key recovery rounds, in a similar manner to what is done in several impossible differential attacks on AES [LDKK08, MDRMH10, BLNS18].

4.5 Optimizing Related Keys

4.5.1 Improving pair generation

Interestingly, in boomerang and impossible boomerang attacks, the pair generation step is independent of the ciphertext value. This means this part is actually independent of the queries, as long as the plaintexts are fixed in advance (and not adaptative). Thus, assuming the differences at the input are the same, the pairs one would make for K_1, K_2 under the key guess $\hat{K}$ are the same as the ones one would make for K_3, K_4 under the key guess $\hat{K} \oplus K_1 \oplus K_3$. This means we only need to compute the pairs on average once per key guess instead of twice.

This optimization is specific to boomerangs with identical input differences in related-key scenarios.

4.5.2 Testing the remaining keys

In a related-key scenario, the boomerang quartets use 4 different keys so the time complexity should be compared to the exhaustive search of the secret key in the same scenario, which is reduced from 2^k to 2^{k-2} as we can test 4 keys at once.

At the same time, the final exhaustive search part of the remaining candidates in the impossible attack can also be done slightly more efficiently than simply testing all surviving keys. Indeed, for similar reasons we can test 4 keys at once. We will do all the tests, except the ones for which all keys have already been eliminated. In general, if we have access to 2^r encryption oracles whose key relations form a vector space of dimension r, and a random key survives the attack with probability p, the exhaustive search part of our key recovery can be done in time

$$2^k \times \frac{1}{2^r}\left(1 - (1-p)^{2^r}\right).$$

This formula quickly converges to p for small values of p, meaning that if we are far away from the exhaustive search cost, we don't gain anything compared to the naive approach. Still, this allows to interpolate between the gain of the filter (p) and the generic

gain of related-key exhaustive search ($1/2^r$). In particular, it gives a gain in cases where the naive approach would fail, for example with $p = 1/e$ and $r = 2$.

This optimization is generic to related-key cryptanalysis under the assumption that the set of key relations is a vector space and that the procedure that eliminates wrong keys removes them independently of the key relation (that is, the probability that a given key is eliminated does not depend on whether its related keys are eliminated).

4.6 Using Multiple Related-Tweak Impossible Differential Distinguishers

For a tweakable block cipher $E_k(M, T)$, we consider here a case where for many non-zero differences $\delta T_{in} \in \Delta T_{in}$ and $\delta T_{out} \in \Delta T_{out}$, we can construct an impossible boomerang distinguisher using $E_k(\cdot, T), E_k(\cdot, T \oplus \delta T_{in}), E_k(\cdot, T \oplus \delta T_{out}), E_k(\cdot, T \oplus \delta T_{in} \oplus \delta T_{out})$, that is, we have many impossible boomerang distinguishers, depending on the tweak differences we choose. As there are more impossibilities here, we can hope for a greater amount of quartets from the same amount of data, which would improve the attack and its data complexity. We note $2^{t_{in}} = |\Delta T_{in}|, 2^{t_{out}} = |\Delta T_{out}|$. We also assume ΔT_{in} and ΔT_{out} do not overlap. We note p_t the probability that a difference $(\delta T_{in}, \delta T_{out})$ allows to produce an impossible distinguisher.

Then, to leverage this degree of freedom, we construct 2^s structures where we take all possible values over Δ_{in}, but also over ΔT_{in} and ΔT_{out}.

Hence, if we neglect the small constants, we use $\mathcal{D} = 2^{s+d_{in}+t_{in}+t_{out}}$ data. From these data, we can construct $\mathcal{P} = 2^{s+2d_{in}+2t_{in}+t_{out}}$ pairs, as the pair elements must have $\delta T_{out} = 0$ (as ΔT_{in} and ΔT_{out} do not overlap). Then, from this pair we can construct $\mathcal{Q} = p_t 2^{2s+4d_{in}+2t_{in}+2t_{out}-2n+2d_{out}}$ quartets. Indeed, a matching quartet must have $\delta T_{in} = 0$ on the other faces, and we can only keep quartets with a matching $(\delta T_{in}, \delta T_{out})$.

Overall, we have $\mathcal{Q} = \mathcal{D}^2 \times 2^{2d_{in}+2d_{out}-2n} p_t$, which is at best (if $p_t = 1$) the same than with a single distinguisher. That is, from a given amount of data, we cannot make more quartets than before, meaning the data complexity cannot be significantly improved this way.

However, this does not mean such a set of impossible boomerang distinguishers is useless, for multiple reasons:

- It increases the maximal amount of data that we can use to make quartets, and this is more efficient than adding a constant in the tweak and using the same distinguisher multiple times,
- With a fixed amount of data, it allows to reduce the number of plaintexts per tweak, with a limited impact on the complexity,
- As the same plaintexts are encrypted under multiple tweaks, parts of the partial decryptions are independent of the tweak and can be done once for all the considered tweaks.

4.7 Experimental Validation of the Key-Recovery

We developed a Python proof-of-concept of the key-recovery procedure presented in Section 4.3. We considered a toy version of Simon with a 16-bit state and 32-bit key. We found a 15-round related-key impossible boomerang distinguisher, with fixed input and output differences, and added 3 rounds of key recovery before and after. The experiments matched perfectly with the theory: a wrong key is not discarded during the attack with probability around $1/e$ when considering the full codebook over the 4 related keys.

5 Applications

As an illustration of our two extensions of the impossible boomerang framework, we propose an application to the SIMON family of block ciphers, leveraging the technique detailed in Section 3.2.1, and an application to SKINNYee, applying the idea of the BCT contradiction of Section 3.2.2. The source code of our models is available at:

https://github.com/xbonnetain/boomerang-distinguishers.

5.1 Impossible Boomerang Attacks of Simon

Specification of Simon. SIMON and SPECK are two families of lightweight block ciphers that were proposed by Beaulieu *et al.* in [BSS+15]. We focus here on SIMON, which comes in 10 variants, denoted SIMON-$2n/k$, where $2n$ is the block size and k the key size. The possible sets of parameters are defined in Table 3 while the round function and the key schedule structure are depicted in Figure 6. In each round, the left n-bit state is modified by a quadratic transformation composed of left rotations and an AND operation. The result is XORed to the right n-bit state and to the n-bit round key, and the two halves are swapped. The round keys are computed with a linear key schedule that takes as input the $k = mn$-bit master key and uses a round constant $c \oplus (z_j)_i$ where $c = 2^n - 4$ and $(z_j)_i$ denotes the ith bit of the constant sequence[4] z_j.

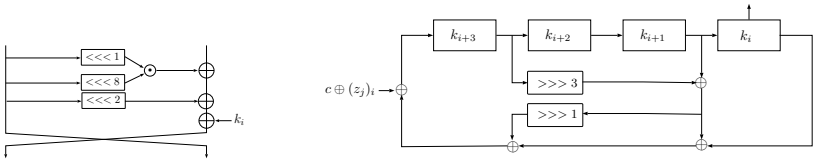

Figure 6: Round function of SIMON (left) and key schedule for the case $m = 4$ key words (right).

Table 3: Parameters of the 10 versions of SIMON.

block size ($2n$)	32	48		64		96		128		
key size (mn)	64	72	96	96	128	96	144	128	192	256
key words (m)	4	3	4	3	4	2	3	2	3	4
const seq	z_0	z_0	z_1	z_2	z_3	z_2	z_3	z_2	z_3	z_4
rounds	32	36	36	42	44	52	54	68	69	72

5.1.1 Models

We propose two models to automatically find impossible boomerang distinguishers for SIMON with an SMT model.

Boomerang incompatibility. Our first approach is to build upon the SMT model for SIMON boomerang distinguishers from [BL23b]. We added support for impossible boomerang distinguishers by considering only probability-1 transitions and adding the constraint that one of the boomerang round transitions had to be impossible. We also extended the model to support all 10 variants of SIMON.

[4] We refer to [BSS+15] for the details of these values.

`Unsat` model. The other approach we propose is to simply model the cipher in SMT, fix the input and output differences of the boomerang distinguisher together with the key differences and try to find a (message, key) pair that fulfills this boomerang configuration. If the solver finds this is `unsat`, then we have an impossible boomerang distinguisher. Some works on impossible differentials had a similar approach: [ST17] searches whether a characteristic that connects the input and output differential exists, and [HLJ+20] models differential propagation in states.

Our approach has the advantage of being able to exactly capture any impossible boomerang, even if the incompatibility stems from finer constraints like key-schedule relations that are not seen if we only look at the characteristics. However, we need to check separately for each difference pair $(\Delta_{in}, \Delta_{out})$, and we restricted ourselves to weight-1 differences. We also managed to make the model work in the related-key setting by forcing round key differences of the form $(\Delta_{in}, 0, \ldots, 0)$ at the beginning and $(0, \ldots, 0, \Delta_{out})$ at the end. While such approach does not cover all possible related-key contradictions, it maximizes the number of blank rounds. Experimentally, this approach did not find any boomerang distinguisher that was not found with the previous approach, but the model is extremely simple and the solving was quite fast, less than 1 core-hour for each instance except the 128-bit versions in related-key. We also obtained some partial related-key results for Simon-128/128 and Simon-128/256.

Model validation. Our approach heavily relies on the soundness of both the models we wrote and the SMT solver. We validated the models using the following approaches:

- All Simon-32/64 distinguishers were experimentally validated on the full codebook on 64 random keys.
- We cross-checked some distinguishers between the models: we can force the differences from the output of one model in the other, and see if the result matches. Of course an `unsat` instance might not correspond to an incompatibility, but experimentally this was the case. Unfortunately we could not cross-check the related-key incompatibility outputs using the `unsat` model, as it was too slow.
- As the `unsat` model is mostly an implementation in SMT of Simon, we validated it using the Simon test vectors.

5.1.2 Distinguishers and Attack on Simon

Distinguishers. We applied our model to the 10 variants of Simon to find impossible boomerang distinguishers in three different scenarios: the single key case, the related-key case and the rotational-xor related key case (which, as a reminder, relies on relations of the form $(x, (x \lll \lambda) \oplus \alpha)$ instead of difference relations, see [AL16]). Our results with the incompatibility model are summarized in Table 4. Results with the `unsat` model are summarized in Table 5.

One of our more promising results in view of the state of the art is on Simon-32/64, for which we obtain impossible boomerang distinguishers for up to 17 rounds. Numerous maximal-length distinguishers exist and we represent a 17-round related-key impossible boomerang distinguisher with a few active bits in α and δ (3 and 1, respectively) in Figure 13 of Appendix A.

Attack. The number of rounds covered by our distinguishers being not competitive with the state of the art of the cryptanalysis for the large variants of Simon (see for instance [LPS21] for a recent overview) we focus on Simon-32/64 and we illustrate the impossible boomerang attack on that variant.

Table 4: Highest number of rounds for which an impossible boomerang distinguisher is found by our model for the 10 variants of SIMON.

Block size	32	48		64		96		128		
Key size	64	72	96	96	128	96	144	128	192	256
Single-key	7	6		7		7		7		
Related-key	17	15	17	16	19	17	20	20	23	25
Rotational-xor related-key	17	15	18	17	20	18	21	21	23	27

Table 5: Results of our `unsat` model. Δ is the shift between the input and output active bit, meaning that for all x, there is an impossible boomerang distinguisher with one active input bit at index x and one active output bit at index $x + \Delta$.† the model was too slow, we did not check all the rotations and some Δ might be missing.

Block size		32	48		64		96		128		
Key size		64	72	96	96	128	96	144	128	192	256
SK	Rounds	7	6		7		7		7		
	Δ	1,15	1,5,9,11,13, 15,19,23		1,7,15,17,25,31		1,7,15, 33,41,47		1,7,15,49,57,63		
RK	Rounds	17	15	17	16	18	16	19	19	-	24
	Δ	1,3,8	4,23	2	3,6,17,22, 23,24,27	2,31	11,37	5	7,11, 53,57†	-	52,56†

We present an attack covering 23 rounds of SIMON-32/64 based on the 17-round related-key impossible boomerang distinguisher depicted in Figure 13. We add 3 rounds at the top and 3 rounds at the bottom for the key recovery. To determine the parameters of the attack, we start by propagating with probability one the difference at the input and at the output of the distinguisher up and down to the plaintext and to the ciphertext, respectively. An illustration of the propagation is given in Figure 7, on which we can see that $d_{in} = 21$ bits and $d_{out} = 11$ bits (corresponding to the light green bits). Note that as the round key is added at the end of the round function we consider equivalent first and last rounds that only correspond to the round key addition, as the other operations do not depend on secret values.

The second important parameter is the number of key bits that an attacker needs to guess in order to check that the input (resp. output) difference of the distinguisher meets the expected differences. These are represented by the small yellow triangles in the figures, and are determined by referring to the properties of the only non-linear operation of the round function, that is of the AND. The differential properties of the AND have been discussed in many papers so we only briefly recall here what guess is required to compute an output difference.

Lemma 1. *To uniquely determine the output difference of the AND of two bits x and y, an attacker needs to guess:*

- *Nothing, if $\Delta x = \Delta y = 0$ (both input bits are inactive),*
- *The value of x if $\Delta x = 0$ and $\Delta y = 1$, respectively the value of y if $\Delta y = 0$ and $\Delta x = 1$ (only one input bit is active),*

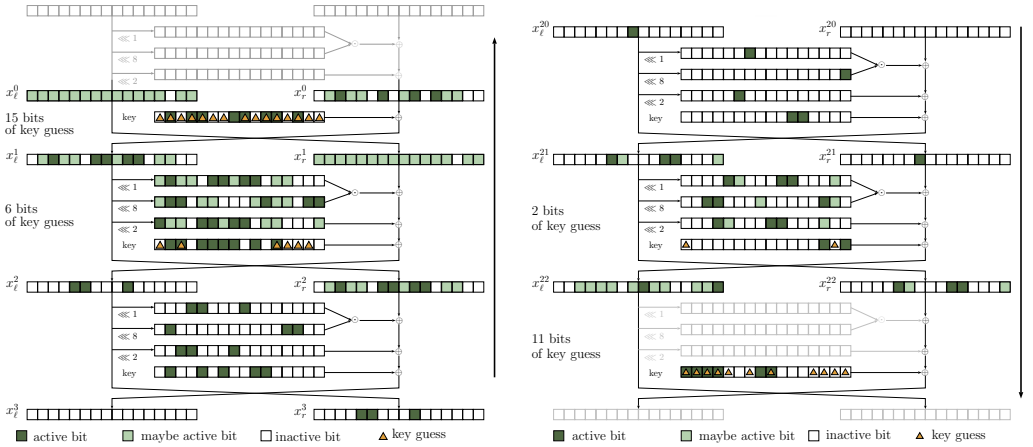

Figure 7: Key recovery rounds for our attack on SIMON-32/64: 3 top rounds (left) and 3 bottom rounds (right). The difference propagation is depicted, together with the key bits that are needed in value to check that the input (resp. output) difference of the distinguisher is met.

- *Whether the value of x is equal to the value of y if $\Delta y = 1$ and $\Delta x = 1$ (both bits are active).*

Proof. The result simply follows from the expression of the output difference of the AND of x and y, given by $xy \oplus (x \oplus \Delta x)(y \oplus \Delta y) = \Delta x \Delta y \oplus y\Delta x \oplus x\Delta y$. □

By following these principles we obtain that $|K_{in}| = 21$ bits and $|K_{out}| = 13$ bits as illustrated in Figure 7.

Now that all the parameters are identified, we can look at the details of the attack complexity. As detailed in Section 4, the first step of the attack is the data generation. Several structures of $2^{d_{in}} = 2^{21}$ messages are created under the keys K_1, K_2, K_3 and K_4, related by the required master key differences. As we consider a single impossible boomerang distinguisher, we will request the full codebook on the 4 keys. The number of structures per key is equal to $2^s = 2^{n-d_{in}} = 2^{11}$. The data complexity is equal to $\mathcal{D} = 4 \times 2^{s+d_{in}} = 2^{34}$.

Following the technique presented in Section 4.3, the pair generation will cost $2^{|K_{in}|} \times 2^{32} \times \frac{4}{23} = 2^{50.5}$. We have to compute for each plaintext encrypted with K_1 its matching plaintext encrypted with K_2 (and identically for K_3 and K_4) according to the first key guesses. This costs 4 rounds of SIMON, as we need to encrypt 2 rounds and then decrypt 2 rounds (we assume here that the attacker computed equivalent plaintexts by peeling off the operations that do not depend on the key, as depicted in Figure 7). Moreover, as the input difference is the same for the two pairs, we can reuse them between key guesses (see Section 4.5.1), meaning each set of pairs can be used for 2 guesses.

We then do the collision step and generate the quartets. This costs

$$2^{|K_{in}|} \times 2^{32\times 2 - 32\times 2 + 2\times d_{out}} = 2^{43} \text{ encryptions},$$

and produces $2^{2d_{out}} = 2^{22}$ quartets per key guess.

We end the impossible search by doing the quartet filtering with early abort. There are 6 active AND at round 21 for which we can compute the output difference in one key guess. Each difference produces a filter of probability $\frac{1}{4}$, as the same key guess is used for the two sides of the boomerang. Thus, even taking into account the additional key guesses,

the total number of quartets vanishes rapidly in the early abort, and this last part has a cost negligible compared to $2^{50.5}$.

Overall, we can expect that a key passes the test with probability $\exp\left(-2^{22} \times 2^{-22}\right) = 1/e$. Hence, the final exhaustive search costs $2^{64} \times \left(1 - (1 - 1/e)^4\right)/4 = 2^{61.75}$ (see Section 4.5.2).

5.1.3 Improved Attack on Simon-32/64

As the round function and key schedule of Simon are rotation-invariant, every differential distinguisher actually defines a family of 16 distinguishers. We unfortunately cannot directly exploit all of them easily as they rely on distinct key relations. However, if an attacker is allowed to query encryptions under multiple key relations, the same data can be reused for several distinguishers and some operations are shared.

For $0 \leq i < 16$, each distinguisher requires access to the keys $(K, K \oplus \Delta_i^1, K \oplus \Delta_i^2, K \oplus \Delta_i^1 \oplus \Delta_i^2)$. This is the affine space $K \oplus \langle \Delta_i^1, \Delta_i^2 \rangle$. Note that the distinguishers will work for any coset of this vector space, that is, with the keys $K \oplus \Delta' \oplus \langle \Delta_i^1, \Delta_i^2 \rangle$, for any Δ'.

Now, we choose two distinguishers, i and j, and consider the affine space of 16 keys $K \oplus \langle \Delta_i^1, \Delta_i^2, \Delta_j^1, \Delta_j^2 \rangle$. In this space we have access to 4 cosets of $\langle \Delta_i^1, \Delta_i^2 \rangle$, and 4 cosets of $\langle \Delta_j^1, \Delta_j^2 \rangle$. Thus, querying encryption under 16 keys offers the possibility to use 2 distinguishers 4 times each.

Key recovery. The key recovery will essentially be 8 independent key recoveries. There is however one part that can be optimized: as the pair generation step only depends on the plaintexts and not on the ciphertexts, the actual pairs do not directly depend on the key, but only on the guessed value for K_{in}. Thus, pairs for a given key guess K_{in} in the distinguisher using $K \oplus \langle \Delta_i^1, \Delta_i^2 \rangle$ can be reused for the copy that use $K \oplus \Delta' \oplus \langle \Delta_i^1, \Delta_i^2 \rangle$ with the key guess $K_{in} \oplus \Delta'$, and similarly for the other copies. Overall, the pair generation step is only done twice and not 8 times.

We consider we have 2^s structures for each distinguisher, which corresponds to $\mathcal{D} = 2^{s+d_{in}+4}$ initial encryption queries. The pair generation costs $2 \times 2^{|K_{in}|} \times 2^{s+d_{in}} \times \frac{4}{23}$ per distinguisher (it is done once per distinguisher, hence the factor 2). Quartet generation costs $2^{|K_{in}|} \times 2^{2s+2d_{in}+2d_{out}-2n}$ and produces $2^{2s+2d_{in}+2d_{out}-2n}$ quartets per key guess, it will be repeated 8 times. As before, the early abort step is of negligible cost and the probability that one key guess survives one of the 8 distinguishers is around $\exp\left(-2^{2s+2d_{in}+2d_{out}-2n-2c_{out}}\right)$.

Full codebook. If we take the full codebook over the 16 keys, the probability a wrong key survives is $\exp(-8)$. Thus, the overall cost is

$$\underbrace{16 \times 2^{32}}_{\mathcal{D}} + \underbrace{2 \times 2^{|K_{in}|} \times 2^{32} \times \frac{4}{23}}_{\mathcal{P}(\times 2)} + \underbrace{8 \times 2^{|K_{in}|} \times 2^{2d_{out}}}_{\mathcal{Q}(\times 8)} + \underbrace{\exp(-8) \times 2^{64}}_{\text{Exhaustive search}} \simeq 2^{53.1}.$$

Note that we did not use the optimization from Section 4.5.2 as it has a negligible impact here.

Reduced data complexity. We can also optimize the attack to obtain a final probability to keep a key comparable to the previous attack. This gives an attack using 2^{10} structures, for an overall data complexity of $2^{10+21+4} = 2^{35}$. Thus, while the data per key can be reduced (only 2^{31} data is encrypted for each key), the overall data cost increases slightly in comparison to the attack proposed in Section 5.1.2. This is due to the quadratic factor that links the number of structures and the final probability. The cost will be dominated by the final exhaustive search, in $2^{59.85}$.

Using more key relations. We could devise a more efficient attack using even more key relations and guessing less bits of K_{in} before making the quartets. However, increasing the number of relations makes the attack less relevant, especially as there are only 64 key bits here.

Alignment of structures. Note that the structures for the two distinguishers intertwine (as they are built upon the same data), and they are not aligned, as the involved bits are rotated. Nonetheless, if we take two distinguishers rotated of 4 bits, they have 6 shared input bits that are on positions of fixed difference (the x_r^0 bits in Figure 7 are $(2,6),(5,9),(6,10),(7,11),(10,14)$ and $(11,15)$, when numbering from left to right). Hence, as long as we encrypt at least 2^{26} data per key, we can assume all the structures are full.

5.2 Impossible Boomerang Attacks of SKINNYee

Specification of SKINNYee. SKINNYee has been introduced by Naito *et al.* at Crypto 2022 [NSS22] as an extension of the SKINNY tweakable block cipher [BJK+16] that would fit their new mode called HOMA. It reuses most of the round function of SKINNY but has a different technique to manage the tweak and the key. The tweakey framework is not used and the tweak and the key remain separated and are added to different parts of the state: the round key on the bottom two rows, and the round tweak on the top two rows (see Figure 8). Other differences from SKINNY are the number of rounds (that is raised to 56) and the details of the round constant generation. Also, since there are 4 tweak states, an additional LFSR, $LFSR_4$, is introduced to supplement $LFSR_2$ and $LFSR_3$ in the tweak schedule. These are defined as follows, where x_0 is the LSB:

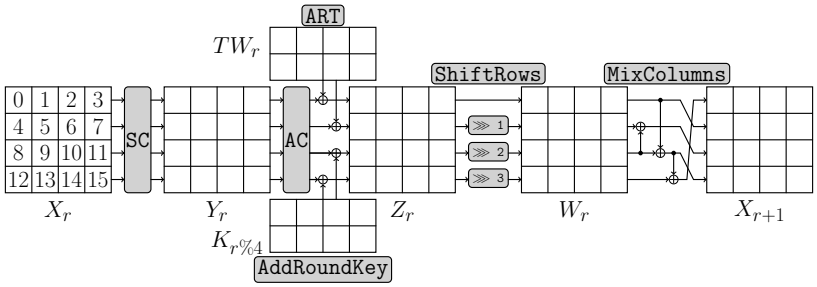

Figure 8: Round function of SKINNYee (picture based on [Jea16]).

$$\begin{aligned} LFSR_2: &\quad (x_3||x_2||x_1||x_0) \quad \rightarrow (x_2||x_1||x_0||x_3 \oplus x_2) \\ LFSR_3: &\quad (x_3||x_2||x_1||x_0) \quad \rightarrow (x_0 \oplus x_3||x_3||x_2||x_1) \\ LFSR_4: &\quad (x_3||x_2||x_1||x_0) \quad \rightarrow (x_1||x_0||x_3 \oplus x_2||x_2 \oplus x_1) \end{aligned}$$

In each round the 4 tweak states are updated by a cell permutation P_T ($P_T = [9,15,8,13,10,14,12,11,0,1,2,3,4,5,6,7]$), and each of the cells of the first two rows of the internal state of state number 2, 3 and 4 is updated with $LFSR_2$, $LFSR_3$ and $LFSR_4$ respectively.

Only one variant of SKINNYee is proposed, with a 64-bit state, a 128-bit key and a 256-bit tweak, completed by 3 bits of domain separator. The authors aim for SKINNYee being a secure tweakable-pseudo-random permutation (TPRP) and exclude the related-key scenario. When used in their mode HOMA, the tweak value is chosen by the mode and is not controlled by an attacker.

5.2.1 Model

We build a word-oriented model to find BCT-based impossible boomerang distinguishers on SKINNYee by adapting the tool for impossible differential distinguishers developed along the article [HSE23]. Our problem is translated into a constraint satisfaction problem (CSP). As a reminder, a CSP is described by 3 sets: X, D and C, that respectively represent the variables, the domain of the variables (that is, the possible values they can take) and the constraints the variables should satisfy.

The model takes as input the number of rounds in the upper and in the lower trails, and looks for an impossible boomerang distinguisher based on a BCT contradiction at the junction of the two trails. It uses variables to represent the differences in the internal state and in the round tweaks of the top trail and of the bottom trail, and we give constraints explaining how these are propagated through the rounds. We require that the setting leading to a BCT contradiction happens, and ask the model if a solution exists for the provided parameters.

We reuse the encoding of [SGWW20, HSE23] to handle the different types of differences in the internal states. The variable `AX[i]` is used to represent the activeness of the cell $X[i]$ ($0 \leq i \leq 15$) and can take 4 values, while `DX[i]` stores the exact difference when it is known:

$$\texttt{AX}[i] = \begin{cases} 0 & \Delta X[i] = 0 \\ 1 & \Delta X[i] \text{ is non zero and known} \\ 2 & \Delta X[i] \text{ can be any nonzero value} \\ 3 & \Delta X[i] \text{ can take any value} \end{cases} \qquad \texttt{DX}[i] \in \begin{cases} \{0\} & \texttt{AX}[i] = 0 \\ \{1, ..., 15\} & \texttt{AX}[i] = 1 \\ \{-1\} & \texttt{AX}[i] = 2 \\ \{-2\} & \texttt{AX}[i] = 3 \end{cases}$$

Constraints are added to ensure that at each time the activity and difference variables associated to the same cell $X[i]$ are consistent:

$$Link(\texttt{AX}[i], \texttt{DX}[i]) := \begin{cases} \text{if } \texttt{AX}[i] = 0 \text{ then } \texttt{DX}[i] = 0 \\ \text{else if } \texttt{AX}[i] = 1 \text{ then } \texttt{DX}[i] > 0 \\ \text{else if } \texttt{AX}[i] = 2 \text{ then } \texttt{DX}[i] = -1 \\ \text{else } \texttt{DX}[i] = -2 \end{cases}$$

These variables are set for each of the 16 nibbles of internal state at step X, Y and Z (following the notations of Figure 8) of the encryption, and independently for both the top and the bottom trail. A series of constraints is then added to model the propagation with probability one of the plaintext difference of the top trail in the encryption direction and of the ciphertext difference of the bottom trail in the decryption direction.

For instance, a cell `X`[i] of known active difference equal to `0x2` (so represented by `AX`[i]=1 and `DX`[i]=2) should become active but of unknown difference after traversing the Sbox layer (`AY`[i]=2 and `DX`[i]=-1). We refer to [HSE23] for a description of these constraints, that we reused as it is.

As there are no differences in the key, only the tweak difference is modeled. We reuse the technique of [HSE23] that takes into account the permutation P_T of the tweak schedule and bounds the number of cancellations. An important difference of our model is that the upper and lower trails are independent, which in particular means that we introduce two tweak states. As detailed in the next section and as one could expect, the trail found by the solver makes good use of this by positioning 3 cancellations in the top trail and 3 in the bottom trail, resulting in many blank rounds.

To force a contradiction at the junction of the upper and lower trail, we add a constraint that states that the input of one of the 16 Sboxes of the last Sbox layer is active and of known value while at the same time the output of the same Sbox in the lower trail is also active and of known value, that is: `((AXU[RU, i] == 1) /\(AYL[0, i] == 1))` (where i is the Sbox index ($0 \leq i \leq 15$) and `RU` is the number of rounds in the upper trail).

Once a (truncated) solution is found by the model, the actual input, output and tweak differences must be set so that the known values appearing in the BCT contradiction actually correspond to a BCT coefficient of value 0. In the distinguisher provided below it corresponds to fixing the master tweak differences so that they develop into a contradicting pair in round 14.

We use the constraint modeling language MiniZinc [NSB+07] and use the CP solver Or-Tools [PF19].

5.2.2 Distinguishers and Attacks on SKINNYee

Distinguisher. We applied the previous model to SKINNYee and found a 21-round impossible boomerang distinguisher, depicted in Figure 14 in Appendix B. Since the boomerang construction allows to have different tweak differences for the top and for the bottom trail it allows to position 8 rounds with no active Sbox at the top *and* at the bottom. (This would not have been possible for an impossible differential distinguisher where only one tweak difference is used and is common to both the top and the bottom trail). Thanks to the linear tweak schedule the attacker knows the exact difference of several words after numerous rounds and can use this to build a BCT contradiction.

Attack. We use the 21-round boomerang impossible distinguisher presented in Figure 14 and add 4 rounds at the top and 4 rounds at the bottom for the key recovery, as depicted in Figure 10 and Figure 11 respectively. As in the case of Simon, the first and last operations do not depend on secret parameters so we can partially encrypt and decrypt the data set before starting the attack to remove the unnecessary operations.

In addition to this, in the first two rounds of the attack we use an equivalent round key (and similarly round tweak) that corresponds to $MC(SR(K_r))$ (similarly $MC(SR(TW_r))$). Into detail, the states corresponding to the key K_r and its equivalent version EK_r are as depicted in Figure 9.

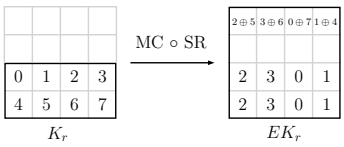

Figure 9: Value of the equivalent key ($EK_r = MC(SR(K_r))$) in function of the nibbles of the key (K_r).

As depicted in Figure 10, the number of active bits at the input is $d_{in} = 5 \times 4 = 20$. The number of key guesses made in EK_0 is equal to $5 \times 4 = 20$ bits (refer to Figure 9 to see the equal nibbles and the positions where no key appears) and in EK_1 $2 \times 4 = 8$ bits need to be guessed, which means that $|K_{in}| = 28$ bits.

Now referring to Figure 11, one can observe that $d_{out} = 12 \times 4 = 48$ bits are active at the ciphertext side. The 17 nibbles of key showed in the figure actually reduce to less as K_0 was already partly guessed in the top part. More into details, for the bottom 4 rounds of key recovery we need nibble 0 to 7 of K_0 (following the numbering of Figure 9), but nibbles 0, 1 and 3 were already guessed, together with the XOR of nibbles 2 and 5 and the XOR of nibbles 3 and 6. Guessing the lower key at once would thus requires 12 nibbles of key guess.

Improvement. The filtering factor ($2^{-2n+2d_{out}}$) appearing when building quartets of messages can actually be further improved by noticing that an additional condition can be checked when pairing pairs together. We refer to Figure 12 for an illustration of the process.

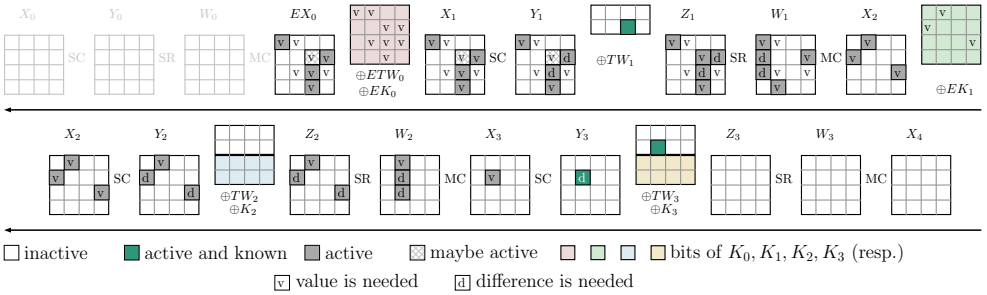

Figure 10: Top 4 rounds added for key recovery in our 29-round attack on SKINNYee. The difference propagation is depicted, together with the nibbles that are needed in value to check that the input difference of the distinguisher is met.

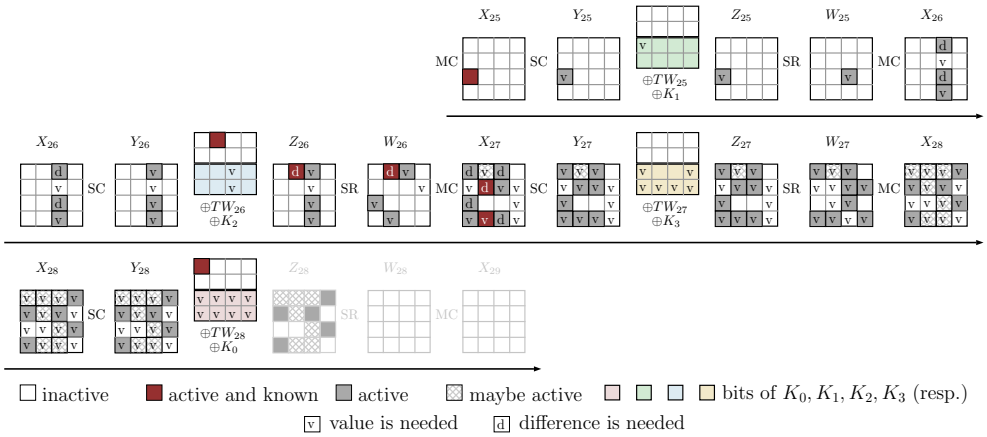

Figure 11: Bottom 4 rounds added for key recovery in our 29-round attack on SKINNYee. The difference propagation is depicted, together with the nibbles that are needed in value to check that the output difference of the distinguisher is met.

We consider that the technique presented in Section 4.3 is followed, and that in particular the attack starts by the construction of pairs of messages thanks to a key guess of the $|K_{in}|$ top key bits. Once this is done, the attacker aims at building quartets of messages by referring to the ciphertext differences of pairs. In our specific case, the attacker can actually rely on the fact that part of the last round key is known (from the top key guess and as the key schedule is very simple) to base this pairing step on a stronger filter.

The additional condition corresponds to the verification that the nibble difference at position 14 of W_{27} is 0 for the internal state corresponding to the pair (C^1, C^3) and to the pair (C^2, C^4). As shown in Figure 12, the attacker has sufficient information to compute $W_{27}[14]$ once the $|K_{in}|$ key bits have been guessed. The idea is thus to compute $W_{27}[14]$ for each message and to store it together with the ciphertext in view of the collision process to build quartets. In addition to checking that the ciphertexts C^1 and C^3 (resp. C^2 and C^4) collide on the $n - d_{out}$ required bits, the attacker thus checks the additional collision on the nibble $W_{27}[14]$, which rises the filter value to $2^{-2n+2d_{out}-2\times 4}$.

Complexity of the attack. The detail of the complexities is as follows. 2^s structures of $2^{d_{in}} = 2^{20}$ messages are queried under 4 related tweaks (chosen among the 103 values that ensure a BCT contradiction). The full codebook is queried under these 4 public values, that is, $2^s = 2^{n-d_{in}} = 2^{44}$, setting the data complexity to $\mathcal{D} = 4 \times 2^{s+d_{in}} = 2^{66}$.

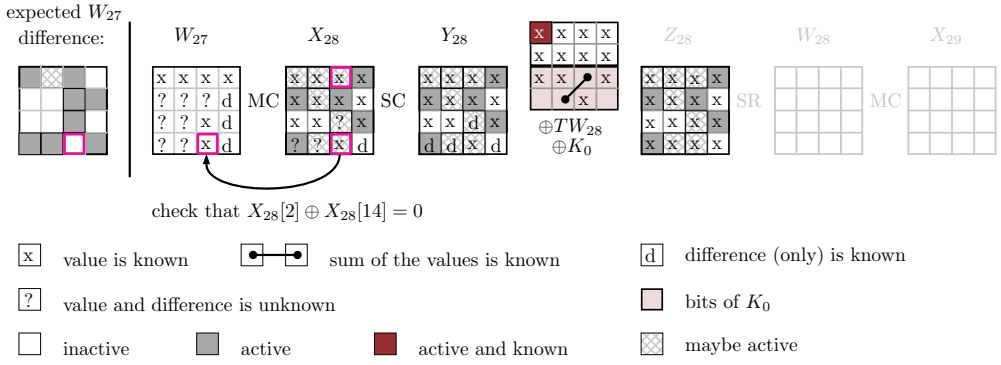

Figure 12: Depiction of the free filter of 4 bits obtained in the last two rounds of the attack.

The pair generation according to the guess of the $|K_{in}| = 28$ top key bits requires a time equivalent to $2^{|K_{in}|} \times 2^n \times 2 \times \frac{8}{29} = 2^{91.14}$ encryptions. The (improved) collision step to create the quartets is then of complexity: $2^{|K_{in}|} \times 2^{64\times2-64\times2+2\times d_{out}-2\times4} = 2^{116}$ and creates $2^{2\times d_{out}-2\times4} = 2^{88}$ quartets per key guess. The final filtering step is done with the early abort technique, where the quartets associated to a specific key guess are gradually decrypted over the last 4 rounds (which requires key guesses from $|K_{out}|$ bits) to check if the difference β is obtained twice, in which case the key guess can be discarded.

In view of the configuration of the 4 last rounds (see Figure 11 and Figure 12), the attacker can for instance start focusing on the first column of X_{28}. A zero-difference is expected in byte 4 and 8 of W_{27}, which implies that the difference in $X_{28}[4]$ and in $X_{28}[12]$ must be equal. Since $Y_{28}[4]$ is known in value and in difference and $Y_{28}[12]$ is known in difference, the attacker can use the following well-known principle to determine the value of $K_0[4]$:

Lemma 2. *For a given Sbox S and given two non-zero differences Δ_a and Δ_b, the equation $S(x \oplus \Delta_a) \oplus S(x) = \Delta_b$ has one solution x on average. The solutions can be efficiently computed.*

Formally, the attacker determines $K_0[4]$ so that: $S^{-1}(Z_{28}[12] \oplus K_0[4]) \oplus S^{-1}(Z_{28}[12] \oplus K_0[4] \oplus \Delta Z_{28}[12]) = \Delta X_{28}[4]$ with the pair (C^1, C^3) and obtains one value on average. Using the second pair (C^2, C^4) they check if this value matches the same requirement, which has a probability of 2^{-4} on average. Doing so the following filtering operations are made on a further reduced amount of quartets, and the rest of the early-abort phase is of negligible cost.

The last step of the attack doing the exhaustive search of the undetermined key bits has a cost of $\frac{2^{128}}{e} = 2^{126.6}$ encryptions.

5.2.3 Improved attack on SKINNYee

Using multiple tweaks. As there are 103 pairs $(\delta T_{in}, \delta T_{out})$ that lead to a contradiction, we can leverage them to obtain a better attack, as explained in Section 4.6. We query 2^s structures of 2^{20} plaintexts, encrypted under 256 tweaks. Thus the data complexity is $\mathcal{D} = 2^{s+d_{in}+8}$. We next follow the pair filtering process and start by guessing $|K_{in}| = 28$ top key bits, and for each guess we can construct $2^{s+d_{in}+12-1}$ pairs. Hence, the cost of this step is $2^{28+11+s+d_{in}} \times \frac{8}{29}$. From these pairs we produce $\frac{103}{256}2^{|K_{in}|+2s+2d_{in}+2d_{out}-2n+16-2} = \frac{103}{256}2^{42+2(s+d_{in}+d_{out}-n)}$ quartets. Overall the probability a wrong key survives is around $\exp\left(-\frac{103}{256}2^{2s+2d_{in}-2n+14}\right)$.

Maximal filtering. We can take $s = n - d_{in} = 44$, as before. Then $\mathcal{D} = 2^{72}$, and the probability that a wrong key survives is around 2^{-9510}. Thus, this is clearly sufficient to eliminate all wrong keys, and arguably too much.

Uniquely identifying the key. We can have the more reasonable aim to uniquely identify the value of $K_{in} \cup K_{out}$. Then we need a probability of 2^{-76}, which can be obtained with $s = 2^{40.5}$ structures. Thus we use $\mathcal{D} = 2^{68.5}$ data overall, the cost of pair generation will be $2^{97.6}$, and the collision will produce overall $2^{121.7}$ quartets. As with the first attack, the early-abort filtering for K_{out} will have a negligible cost. Thus, overall the attack will cost $2^{121.7}$.

Balancing the cost. Similarly, with $s = 39.2$, we can balance the cost of the quartet generation and the final exhaustive search step, with an overall cost of $2^{119.2}$, using $\mathcal{D} = 2^{67.2}$ data overall.

Minimal data complexity. If we want the same filtering probability as in the first attack, in $1/e$, we can take $s = 2^{37.7}$. Thus we use $\mathcal{D} = 2^{65.7}$ data overall, the cost of pair generation will be $2^{94.8}$ and the collision will produce overall $2^{116.1}$ quartets. Finally the dominant cost will be the exhaustive search, in $2^{128}/e \simeq 2^{126.6}$. Interestingly, data complexity is very slightly lower than in the first attack.

6 Conclusion

In this work we extend the notion of impossible boomerang attack from [Lu08] by leveraging the recent advances in the understanding of boomerang incompatibilities. We use these incompatibilities to define new models to find impossible boomerang distinguishers on Simon and SKINNYee. Moreover, we provide a fine analysis of the complexities of such attacks, and apply this framework to Simon and SKINNYee. We obtain a 23-round attack on Simon-32/64 and a 29-round attack on SKINNYee. This last attack breaks two more rounds than the previous best known attack on SKINNYee.

We observed that with impossible boomerangs there is an even starker data threshold than with impossible differentials: for SKINNYee, between an attack that allows to gain 1 bit over exhaustive search and an attack that fully identifies the key, there is only a factor 7 in data.

Acknowledgments

The authors would like to thank the anonymous ToSC reviewers and Fukang Liu for their valuable comments.

This work has been partially supported by the French Agence Nationale de la Recherche through the OREO project under Contract ANR-22-CE39-0015, through the France 2030 program under grant agreement No. ANR-22-PECY-0010 CRYPTANALYSE and it has been partially funded by the European Union (ERC-2023-COG, SoBaSyC, 101125450). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

[AL16] Tomer Ashur and Yunwen Liu. Rotational cryptanalysis in the presence of constants. IACR Trans. Symm. Cryptol., 2016(1):57–70, 2016. https://tosc.iacr.org/index.php/ToSC/article/view/535.

[BBS99a] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In Jacques Stern, editor, EUROCRYPT'99, volume 1592 of LNCS, pages 12–23. Springer, Heidelberg, May 1999.

[BBS99b] Eli Biham, Alex Biryukov, and Adi Shamir. Miss in the middle attacks on IDEA and Khufu. In Lars R. Knudsen, editor, FSE'99, volume 1636 of LNCS, pages 124–138. Springer, Heidelberg, March 1999.

[BDK05] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In Ronald Cramer, editor, EUROCRYPT 2005, volume 3494 of LNCS, pages 507–525. Springer, Heidelberg, May 2005.

[BHL+20] Hamid Boukerrou, Paul Huynh, Virginie Lallemand, Bimal Mandal, and Marine Minier. On the Feistel counterpart of the boomerang connectivity table (long paper). IACR Trans. Symm. Cryptol., 2020(1):331–362, 2020.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, CRYPTO 2016, Part II, volume 9815 of LNCS, pages 123–153. Springer, Heidelberg, August 2016.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In Mitsuru Matsui, editor, ASIACRYPT 2009, volume 5912 of LNCS, pages 1–18. Springer, Heidelberg, December 2009.

[BL23a] Augustin Bariant and Gaëtan Leurent. Truncated boomerang attacks and application to AES-based ciphers. In Carmit Hazay and Martijn Stam, editors, EUROCRYPT 2023, Part IV, volume 14007 of LNCS, pages 3–35. Springer, Heidelberg, April 2023.

[BL23b] Xavier Bonnetain and Virginie Lallemand. On boomerang attacks on quadratic Feistel ciphers new results on KATAN and Simon. IACR Trans. Symm. Cryptol., 2023(3):101–145, 2023.

[BLNS18] Christina Boura, Virginie Lallemand, María Naya-Plasencia, and Valentin Suder. Making the impossible possible. Journal of Cryptology, 31(1):101–133, January 2018.

[BNS14] Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In Palash Sarkar and Tetsu Iwata, editors, ASIACRYPT 2014, Part I, volume 8873 of LNCS, pages 179–199. Springer, Heidelberg, December 2014.

[BSS+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. SIMON and SPECK: Block ciphers for the internet of things. Cryptology ePrint Archive, Report 2015/585, 2015. https://eprint.iacr.org/2015/585.

[CCW+18] Zhihui Chu, Huaifeng Chen, Xiaoyun Wang, Xiaoyang Dong, and Lu Li. Improved integral attacks on SIMON32 and SIMON48 with dynamic key-guessing techniques. Secur. Commun. Networks, 2018:5160237:1–5160237:11, 2018.

[CDJ+20] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas-López, Mridul Nandi, and Yu Sasaki. ESTATE: A lightweight and low energy authenticated encryption mode. IACR Trans. Symm. Cryptol., 2020(S1):350–389, 2020.

[CHP+18] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang connectivity table: A new cryptanalysis tool. In Jesper Buus Nielsen and Vincent Rijmen, editors, EUROCRYPT 2018, Part II, volume 10821 of LNCS, pages 683–714. Springer, Heidelberg, April / May 2018.

[CW16] Huaifeng Chen and Xiaoyun Wang. Improved linear hull attack on round-reduced simon with dynamic key-guessing techniques. In Thomas Peyrin, editor, FSE 2016, volume 9783 of LNCS, pages 428–449. Springer, Heidelberg, March 2016.

[DDV20] Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the fastest boomerangs application to SKINNY. IACR Trans. Symm. Cryptol., 2020(4):104–129, 2020.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir. A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. In Tal Rabin, editor, CRYPTO 2010, volume 6223 of LNCS, pages 393–410. Springer, Heidelberg, August 2010.

[DKS14] Orr Dunkelman, Nathan Keller, and Adi Shamir. A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. Journal of Cryptology, 27(4):824–849, October 2014.

[HBS21] Hosein Hadipour, Nasour Bagheri, and Ling Song. Improved rectangle attacks on SKINNY and CRAFT. IACR Trans. Symm. Cryptol., 2021(2):140–198, 2021.

[HKLP05] Seokhie Hong, Jongsung Kim, Sangjin Lee, and Bart Preneel. Related-key rectangle attacks on reduced versions of SHACAL-1 and AES-192. In Henri Gilbert and Helena Handschuh, editors, FSE 2005, volume 3557 of LNCS, pages 368–383. Springer, Heidelberg, February 2005.

[HLJ+20] Xichao Hu, Yongqiang Li, Lin Jiao, Shizhu Tian, and Mingsheng Wang. Mind the propagation of states - new automatic search tool for impossible differentials and impossible polytopic transitions. In Shiho Moriai and Huaxiong Wang, editors, ASIACRYPT 2020, Part I, volume 12491 of LNCS, pages 415–445. Springer, Heidelberg, December 2020.

[HSE23] Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In Carmit Hazay and Martijn Stam, editors, EUROCRYPT 2023, Part IV, volume 14007 of LNCS, pages 128–157. Springer, Heidelberg, April 2023.

[Jea16] Jérémy Jean. TikZ for Cryptographers. https://www.iacr.org/authors/tikz/, 2016.

[Kir15] Aleksandar Kircanski. Analysis of boomerang differential trails via a SAT-based constraint solver URSA. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, ACNS 15, volume 9092 of LNCS, pages 331–349. Springer, Heidelberg, June 2015.

[Knu98] Lars Knudsen. Deal-a 128-bit block cipher. complexity, 258(2):216, 1998.

[LDKK08] Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim. New impossible differential attacks on AES. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, INDOCRYPT 2008, volume 5365 of LNCS, pages 279–293. Springer, Heidelberg, December 2008.

[LPS21] Gaëtan Leurent, Clara Pernot, and André Schrottenloher. Clustering effect in simon and simeck. In Mehdi Tibouchi and Huaxiong Wang, editors, ASIACRYPT 2021, Part I, volume 13090 of LNCS, pages 272–302. Springer, Heidelberg, December 2021.

[Lu08] Jiqiang Lu. Cryptanalysis of block ciphers. PhD thesis, University of London UK, 2008.

[Lu11] Jiqiang Lu. The (related-key) impossible boomerang attack and its application to the AES block cipher. Des. Codes Cryptogr., 60(2):123–143, 2011.

[LWL22] Chenmeng Li, Baofeng Wu, and Dongdai Lin. Generalized boomerang connectivity table and improved cryptanalysis of GIFT. In Yi Deng and Moti Yung, editors, Information Security and Cryptology - 18th International Conference, Inscrypt 2022, Beijing, China, December 11-13, 2022, Revised Selected Papers, volume 13837 of Lecture Notes in Computer Science, pages 213–233. Springer, 2022.

[MDRMH10] Hamid Mala, Mohammad Dakhilalian, Vincent Rijmen, and Mahmoud Modarres-Hashemi. Improved impossible differential cryptanalysis of 7-round AES-128. In Guang Gong and Kishan Chand Gupta, editors, INDOCRYPT 2010, volume 6498 of LNCS, pages 282–291. Springer, Heidelberg, December 2010.

[Mur11] Sean Murphy. The return of the cryptographic boomerang. IEEE Trans. Inf. Theory, 57(4):2517–2521, 2011.

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

[NSS22] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Secret can be public: Low-memory AEAD mode for high-order masking. In Yevgeniy Dodis and Thomas Shrimpton, editors, CRYPTO 2022, Part III, volume 13509 of LNCS, pages 315–345. Springer, Heidelberg, August 2022.

[PF19] Laurent Perron and Vincent Furnon. Or-tools. https://developers.google.com/optimization, 2019.

[PT22] Thomas Peyrin and Quan Quan Tan. Mind your path: On (key) dependencies in differential characteristics. IACR Trans. Symm. Cryptol., 2022(4):179–207, 2022.

[QHS15] Kexin Qiao, Lei Hu, and Siwei Sun. Differential security evaluation of simeck with dynamic key-guessing techniques. Cryptology ePrint Archive, Report 2015/902, 2015. https://eprint.iacr.org/2015/902.

[SGWW20] Ling Sun, David Gerault, Wei Wang, and Meiqin Wang. On the usage of deterministic (RK) TDs and MDLAs for SPN ciphers. IACR Trans. Symm. Cryptol., 2020(3):262–287, 2020.

[SQH19] Ling Song, Xianrui Qin, and Lei Hu. Boomerang connectivity table revisited. IACR Trans. Symm. Cryptol., 2019(1):118–141, 2019.

[ST17] Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, EUROCRYPT 2017, Part III, volume 10212 of LNCS, pages 185–215. Springer, Heidelberg, April / May 2017.

[Wag99] David Wagner. The boomerang attack. In Lars R. Knudsen, editor, FSE'99, volume 1636 of LNCS, pages 156–170. Springer, Heidelberg, March 1999.

[WP19] Haoyang Wang and Thomas Peyrin. Boomerang switch in multiple rounds. IACR Trans. Symm. Cryptol., 2019(1):142–169, 2019.

[WSW+23] Libo Wang, Ling Song, Baofeng Wu, Mostafizar Rahman, and Takanori Isobe. Revisiting the boomerang attack from a perspective of 3-differential. IEEE Transactions on Information Theory, 2023.

[WWS23] Dachao Wang, Baocang Wang, and Siwei Sun. SAT-aided automatic search of boomerang distinguishers for ARX ciphers. IACR Trans. Symm. Cryptol., 2023(1):152–191, 2023.

[YSS+22] Qianqian Yang, Ling Song, Siwei Sun, Danping Shi, and Lei Hu. New properties of the double boomerang connectivity table. IACR Trans. Symm. Cryptol., 2022(4):208–242, 2022.

[ZDM+20] Boxin Zhao, Xiaoyang Dong, Willi Meier, Keting Jia, and Gaoli Wang. Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and GIFT. Des. Codes Cryptogr., 88(6):1103–1126, 2020.

A 17-round related-key impossible boomerang distinguisher of Simon-32/64

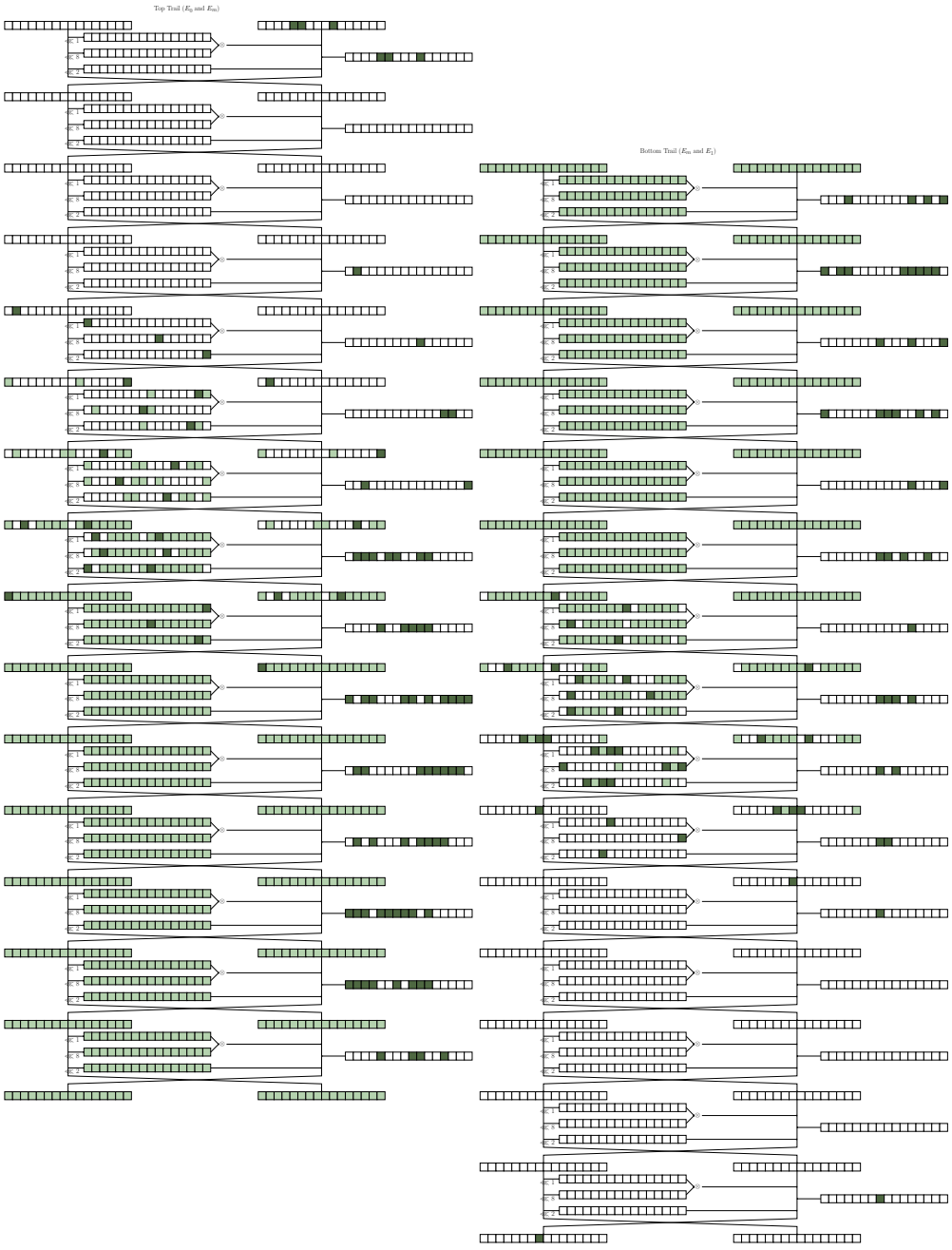

Figure 13: Related-key impossible boomerang distinguisher of 17-round Simon-32/64. White cells are inactive bits, light colored ones are undetermined and darker ones are active.

B 21-round impossible boomerang distinguisher of SKINNYee

Figure 14: 21-round impossible boomerang distinguisher of SKINNYee.