# Analysis of Boomerang Differential Trails via a SAT-Based Constraint Solver URSA

Aleksandar Kircanski
1495E 28th St, 4B, Brooklyn, NY, 11229, US

**Abstract**

In order to obtain differential patterns over many rounds of a cryptographic primitive, the cryptanalyst often needs to work on local differential trail analysis. Examples include merging two differential trail parts into one or, in the case of boomerang and rectangle attacks, connecting two short trails within the quartet boomerang setting. In the latter case, as shown by Murphy in 2011, caution should be exercised as there is increased chance of running into contradictions in the middle rounds of the primitive. In this paper, we propose the use of a SAT-based constraint solver URSA as aid in analysis of differential trails and find that previous rectangle/boomerang attacks on XTEA and SHACAL-1 block ciphers and SM3 hash function are based on incompatible trails. Given the C specification of the cryptographic primitive, verifying differential trail portions requires minimal work on the side of the cryptanalyst.

## 1 Introduction

Differential cryptanalysis [6, 52] is a technique used to break cryptographic primitives such as block ciphers or hash functions. It rests on the existence of high-probability differential trails. A differential trail for an iterative cryptographic primitive can be seen as a sequence of constraints modeling the relations between inner states of primitive executions [29, 25]. Differential trails are built either manually [52, 51, 54], or, with the help of automated tools [9, 38, 30]. To estimate the overall probability of a given differential trail, certain independence assumptions between the constraints need to be introduced.

The validity of such independence assumptions may not always be justified as the constraints may interact and such interactions may severely influence the overall probability calculation. This is especially the case when differential analysis is used to model quartets of primitive executions as opposed to pairs. For example, in the context of boomerang or rectangle attacks, two short high-probability differential trails are connected in one differential pattern over many rounds of the primitive [48, 4].

In 2011, Murphy provided examples of boomerang differential trails that impose dependent constraints on the AES and DES S-boxes [42]. When the dependencies are taken into account, the probability of the overall probabilistic pattern drops to 0. Subsequently, several previously used boomerang trails for primitives based on the Addition, Rotation and Xor (ARX) [45] operations were found to be *incompatible*, i.e., have the probability equal to 0. For example, this was the case for boomerang differential attacks against BLAKE [8] and Skein [10], which

invalidated the corresponding attacks [30]. The discussion related to Murphy's initial doubts [42] was continued by Kim *et al.* in [26]. It was argued that the only reliable way to estimate the boomerang/rectangle attack probability is to attempt to perform the attack itself. Since this is often impossible due to the high computational complexity requirements, estimating the probabilities or their lower bounds via independence assumptions often remains the only way to assess the attack success rate (see, e.g., [3]).

In general, the compatibility or incompatibility of a set of differential constraints can be established as follows. Given a set of constraints, one can simply attempt to find particular inputs for the cryptographic primitive that will conform to such a constraint set in the given round/step span, using techniques such as such as message modification [52]. The main drawback of this approach is that it requires custom implementations and potentially tedious work, e.g., when attempting to prove that some particular boomerang trails are incompatible. Another way to establish (in)compatibility is to apply differential constraint reasoning, where one abstracts away from particular inner state bit-values and deduces consequences from the current differential knowledge base. In case of ARX primitives, one-bit and also multi-bit constraints have been proposed for such reasoning [9, 36, 30]. In 2012, a tool for reasoning on arbitrary ARX primitives using multi-bit constraints has been proposed by Leurent [30]. Although very powerful, ARXtools also has some limits when it comes constraint compatibility verification. Namely, the primitive specification may be somewhat cumbersome and also analysis of primitives with non-ARX components is not possible.

There is a large body of previous work in the area of applying SAT solvers for the purpose of cryptanalysis. This was done for a wide variety of cryptographic primitives, some of which are DES, MD4/5, Trivium, AES and Keccak [35, 21, 13, 23, 40]. One of the tools used in this area is CryptoMiniSat [47]. More powerful theories (than predicate logic) and solvers were also used in cryptanalysis, including a constraint solver STP [14]. Examples include establishing probability upper bounds for differential trails in the case of Salsa20 stream cipher and NORX scheme for authenticated encryption scheme [22, 41]. Closely related to our work are [39, 44], while [44] was developed parallel to our work.

In 2012, a SAT-based constraint solver URSA (Uniform Reduction to SAt) was proposed [20]. It simplifies using SAT solvers in tasks such as cryptanalysis problems. Namely, instead of encoding a problem directly in terms of propositional formulae, the user has to specify the problem in a custom, C-like specification language. In many situations, this means that the C implementation of cryptographic algorithms can be directly used by the URSA system.

**Our contribution:** We show that using the URSA system in conjunction with SAT solvers represents an easy-to-use asset for the cryptanalyst working on local analysis of differential trails. Using this approach, we analyze best previous rectangle attacks on the XTEA and SHACAL-1 block ciphers and locate contradictions in these trails. In addition, we detect contradictions in the trails used in the boomerang distinguisher reaching the highest number of rounds of the SM3 hash function. This shows that the probability estimations for these attacks are invalid and that it remains unknown whether the attacks will work or not. Next, using the URSA system, we find examples of *unaligned* rectangle trails in the context of XTEA block cipher (end of Section 3.1). The existence of such trails has been mentioned previously in [4] and it is interesting to note actual example of such trails can be found by a SAT solver. Finally, we point out a type of contradiction that occurs in primitives with linear key/message expansions which was not discussed in previous literature and suggest that boomerang and rectangle attacks should be verified against this type of contradiction (Section 3.2).

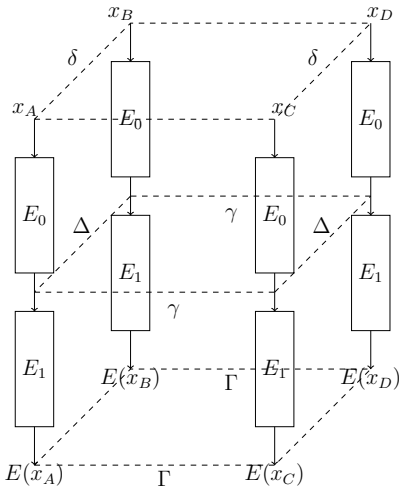


Figure 1: The Rectangle Attack

This paper is organized as follows. In Section 2 we review the the rectangle attack, reasoning on bit-constraints, the URSA system, and also present the notation used throughout the paper. The incompatibilities found in the rectangle trails for XTEA and SHACAL-1 are discussed in Sections 3.1 and 3.2. Finally, the analysis of boomerang trails used in the SM3 distinguisher is given in Section 3.3. The conclusion is provided in Section 4.

# 2 Background and Notation

In this section, a brief description of the rectangle attacks on block ciphers and boomerang distinguishers on hash functions is provided, followed by an introduction to 1-bit conditions and reasoning about differential trails. Finally, an overview of the URSA system is provided along with the notation used throughout the paper.

## 2.1 The Rectangle Attack

In 1999, Wagner introduced a chosen-ciphertext cryptanalytic technique against block ciphers and named it the boomerang attack [48]. The technique exploits non-random behavior of carefully crafted encryption quartets. It works well against ciphers for which there exist short differentials with very high probability. The amplified boomerang attack [24], also known as the rectangle attack [4], is a chosen-plaintext variant of the boomerang attack.

Below, a rectangle attack against a cryptographic function such as a block cipher is summarized. Denote the generic permutation in question by $E$ and it's input by $x$. The quartet structure that the adversary is interested in is shown in Fig. 1. The function is decomposed as $E = E_1 \circ E_0$ and two differential trails are assumed to exist: $\delta \xrightarrow{E_0} \Delta$ and $\gamma \xrightarrow{E_1} \Gamma$ with probabilities $p$ and $q$, respectively. Here, $\delta$ and $\gamma$ are the input differences for $E_0$ and $E_1$, respectively and $\Delta$ and $\Gamma$ are the output differences. If the differentials propagate as specified in Fig. 1, the quartet is called a *right* quartet.

The main idea in the rectangle attack is to compute pairs of the form $(E(x_A), E(x_A \oplus$

$\delta$)) for many randomly chosen $x_A$ inputs and to count how many *pairs* of such of pairs will constitute right quartets. The probabilistic analysis of such an event is as follows. Out of $N$ encrypted pairs with input difference $\delta$, about $p \cdot N$ will conform to the $\delta \xrightarrow{E_0} \Delta$ trail. Now, out of $p \cdot N$ such pairs, one can have about $\frac{(p \cdot N)^2}{2}$ candidate quartets. The probability that $E_0(x_A) \oplus E_0(x_C) = \gamma$ within a randomly chosen candidate quartet is $2^{-n}$, where $n$ is the $E_0$ output bit-length. This event actually always coincides with $E_0(x_B) \oplus E_0(x_D) = \gamma$ since $E_0(x_B) \oplus E_0(x_D) = E_0(x_A) \oplus \delta \oplus E_0(x_C) \oplus \delta = \delta \oplus \delta \oplus \gamma = \gamma$ and thus the probability of both $E_0(x_A) \oplus E_0(x_C) = E_0(x_B) \oplus E_0(x_D) = \gamma$ is in fact $2^{-n}$. As a result, the expect number of quartets satisfying $E(x_A) \oplus E(x_C) = E(x_A) \oplus E(x_C) = \Gamma$ is $\frac{(p \cdot N)^2}{2} \cdot 2^{-n} \cdot q^2 = N^2 \cdot 2^{-n-1} \cdot p^2 \cdot q^2$.

The expected number of right quartets is augmented further by allowing the two differential trails to vary, i.e., by considering $\delta \xrightarrow{E_0} \Delta'$ and $\gamma' \xrightarrow{E_1} \Gamma$ for all possible valid pair choices for $(\Delta', \gamma')$, that is

$$N^2 \cdot 2^{-n-1} \cdot \sum_{(\Delta', \gamma')} p^2_{\delta \to \Delta'} \cdot q^2_{\gamma' \to \Gamma} \tag{1}$$

On the other hand, for a random permutation, the expected number of right quartets is $\frac{N^2}{2} \cdot 2^{-2n} = N^2 \cdot 2^{-2n-1}$. Comparing this estimate to (1) yields that if $\sum_{(\Delta', \gamma')} p^2_{\delta \to \Delta'} \cdot q^2_{\gamma' \to \Gamma} \gg 2^{-n}$, $E$ can be distinguished from a random permutation.

In the literature [4, 32, 33, 34], the estimate (1) is further simplified as

$$N^2 \cdot 2^{-n-1} \cdot \sum_{\Delta'} p^2_{\delta \to \Delta'} \cdot \sum_{\gamma'} q^2_{\gamma' \to \Gamma}$$

which is a sound estimate if one assumes the pairwise independence of all $E_0$ and $E_1$ trails.

As for the boomerang distinguisher for hash functions, the goal is to is to find a quartet $(x_0, x_1, x_2, x_3)$ for function $f$ such that

$$\begin{gathered} x_0 \oplus x_1 \oplus x_2 \oplus x_3 = 0 \\ f(x_0) \oplus f(x_1) \oplus f(x_2) \oplus f(x_3) = 0 \end{gathered} \tag{2}$$

which is called a *zero-sum* or equivalently, a *second-order collision*. This is done by a technique similar to the above described distinguisher, taking into account the message freedom that is available in the context of compression functions. For a more detailed introduction to boomerang distinguishers on hash functions, the reader is referred to [7].

## 2.2 Reasoning on 1-bit constraints

Searching for differential trails is facilitated by a constraints language introduced in [9]. Instead of working with *bit-values*, reasoning is performed on *bit-constraints*. The symbols used for expressing bit-constraints are provided in Table 1. For example, when we write `-x-u`, we mean a set of 4-bit pairs

$$\texttt{-x-u} = \{T, T' \in F_2^4 | T_3 = T'_3, T_2 \neq T'_2, T_1 = T'_1, T_0 = 0, T'_0 = 1\}$$

where $T_i$ denotes $i$-th bit in word $T$.

| $\delta(x, x')$ | meaning | (0,0) | (0,1) | (1,0) | (1,1) |
|---|---|---|---|---|---|
| ? | anything | √ | √ | √ | √ |
| - | $x = x'$ | √ | - | - | √ |
| x | $x \neq x'$ | - | √ | √ | - |
| 0 | $x = x' = 0$ | √ | - | - | - |
| u | $(x, x') = (0, 1)$ | - | √ | - | - |
| n | $(x, x') = (1, 0)$ | - | - | √ | - |
| 1 | $x = x' = 1$ | - | - | - | √ |
| # | | - | - | - | - |

| $\delta(x, x')$ | meaning | (0,0) | (0,1) | (1,0) | (1,1) |
|---|---|---|---|---|---|
| 3 | $x = 0$ | √ | √ | - | - |
| 5 | $x' = 0$ | √ | - | √ | - |
| 7 | | √ | √ | √ | - |
| A | $x' = 1$ | - | √ | - | √ |
| B | | √ | √ | - | √ |
| C | $x = 1$ | - | - | √ | √ |
| D | | √ | - | √ | √ |
| E | | - | √ | √ | √ |

Table 1: Symbols used to express 1-bit conditions [9]

Next, small examples of (a) a differential trail (b) a boomerang trail and (c) a boomerang trail incompatibility are provided. As for the differential trail, consider the following constraint specification over one 4-bit modular addition

$$\texttt{----} + \texttt{---x} = \texttt{---x} \tag{3}$$

The trail models a pair of additions $x_A + y_A = z_A$ and $x_B + y_B = z_B$ and specifies that $x_A = x_B$ and also that $y_A$ and $y_B$, as well as $z_A$ and $z_B$ are different only on the least significant bit. It can be observed that is the necessary condition for the trail to realize is $lsb(x_A) = lsb(x_B) = 0$.

As for the boomerang trail, in that context, one works with quartets instead of pairs. Consider a quartet of modular additions $x_\omega + y_\omega = z_\omega$, for $\omega \in \{A, B, C, D\}$. To specify a boomerang trail, two differential trails are required, labeled as the *top* trail and the *bottom* trail. This terminology comes from the fact that the two trails are specified on the bottom and the top round portions of the cryptographic primitive, respectively. For the purpose of this example, let (3) be the bottom trail and let the top trail be specified by

$$\texttt{----} + \texttt{---x} = \texttt{--xx} \tag{4}$$

The bottom trail is imposed on $x_\omega + y_\omega = z_\omega$ for $\omega \in \{A, B\}$ and $\omega \in \{C, D\}$, whereas the bottom trail for $\omega \in \{A, C\}$ and $\omega \in \{B, D\}$. As shown below, taking the four sets of constraints on $x_\omega + y_\omega = z_\omega$ for $\omega \in \{A, B, C, D\}$ yields a contradiction, i.e., the boomerang trail incompatibility. The incompatibility of (3) and (4) follows from the fact the the necessary condition to have (4) is that the rightmost bit of $x$ needs to equal to 1, i.e., that $lsb(x_A) = lsb(x_C) = 1$ (and also $lsb(x_B) = lsb(x_D) = 1$). However, as shown above, the necessary condition for (3) is that $lsb(x_A) = lsb(x_B) = 0$ and thus no quartet of additions satisfy both trails in the described sense.

## 2.3 The URSA System

The system was proposed in 2012 [20] and represents a high-level front-end to efficient SAT solves. It translates constraints sets specified in C-like language into SAT formulas, after which a SAT solver of user's preference is run on the derived equations. There are two variable types in the URSA language: (unsigned) numerical with names of variables starting with `'n'` and boolean with variable names starting with `'b'`. All arithmetic operations over numeric variables are performed modulo $n$, where $n$ is user-specified parameter. In URSA, there are control flow structures (such as `for` and `if`) as well as procedures and C-like arrays.

The URSA approach can be illustrated by the following example taken from [20]. Consider the problem of finding $x_0$ given $x_{100}$ for the recurrence relation specified by $x_{n+1} = (1664525x_n + 1013904223) \bmod 2^{32}$. The URSA code that corresponds to this problem is

```
nx=nseed;
for (ni=1; ni<=100; ni++)
     nx = nx*1664525+1013904223;
bc = (nx == 3998113695);
assert(bc);
```

The code above is processed by running `cat rec.ursa | ursa -l32`, where the variable bit-length $n = 32$ is specified in the command line. The system will solve for the independent variable `nseed` and return it's value.

Another example of the URSA constraint specification is given in the Appendix. The provided example corresponds to the (incompatible) boomerang trails example discussed in Section 2.2. For more details about the URSA system, see [20].

### 2.4 Notation

The following notation is used throughout the paper:

$x^b$: The $b^{th}$ bit of a word $x$. For example $x^0$ is the least significant bit of $x$.
$A, B, C, D$: four branches of primitive executions, following Fig. 1.
$\Delta r_j^i[A, B]$: bit-constraint (a symbol from Table 1) at bit-position $i$ in word $r_j$ constraining branches $A$ and $B$.
$\oplus$, $+$: bit-wise XOR and addition mod $2^{32}$, respectively
$\ll$, $\gg$: left and right shift, defined on 32-bit values.
$\lll$, $\ggg$: left and right rotation, defined on 32-bit values.

## 3 Detecting Rectangle/Boomerang Trail Contradictions

In this section, we detect contradictions in the trails used in attacks on XTEA [31], SHACAL-1 [49, 12] block ciphers and the SM3 [3] hash function. The first two attacks are rectangle related-key key recovery attacks and the latter attack is a distinguishing attack against a reduced-round SM3 compression function.

The general approach is to represent the primitive and the corresponding step constraints in the URSA language, run a SAT solver over the sequence of steps where a contradiction is suspected, i.e., typically around the middle steps where the rectangle trail switch [48, 24, 4] occurs. If the SAT solver reports no solutions, the next step is to locate where the contradiction is located, i.e., to find the minimal or close to minimal constraint set that yields a contradiction. This was done using a manual trial-and-error approach, i.e., by removing constraints as long as the system does not have solutions. Finally, the proof for the contradiction is built based on the reduced constraint set.

### 3.1 On the incompatibility of XTEA trails [31]

The key-recovery attack on 36-reduced-round XTEA [31] is a related-key attack since it requires differences in the key bits (as well as in plaintexts). It works with quartets of encryptions and

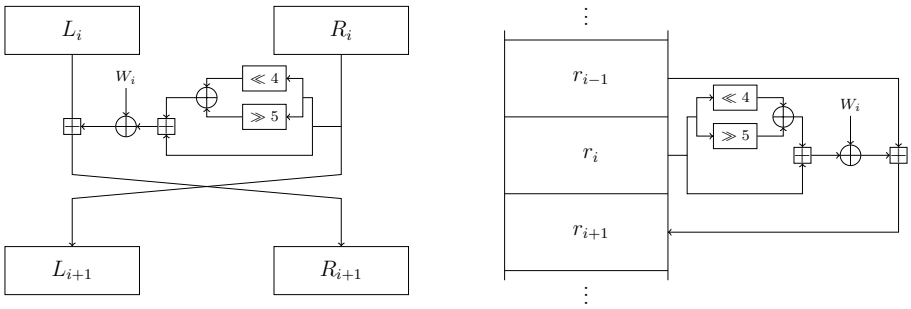


Figure 2: Two equivalent representations of the XTEA round function

falls into the category of rectangle attacks. Below, a brief specification of the cipher is provided. For a more detailed description, the reader is referred to [53, 31].

XTEA takes as input a 64-bit plaintext and a 128-bit key. The encryption and decryption functions consist of 64 Feistel-network rounds. Two equivalent representations of one encryption round are schematically presented in Fig. 2, where on the right-hand side a shift-register based representation is provided. Feistel networks have been first studied in the form of shift registers in the context of the DES block cipher [15], where the cipher was presented as a Non-Linear Feedback Shift Register with input. We use the shift-register based representations since such representations provide are elegant when it comes to working with differential paths [9, 38].

The 128-bit key is represented by four 32-bit words as $K = (K_0, K_1, K_2, K_3)$ and then, for $i = 1, \ldots 64$ expanded to 64 32-bit words, as specified by

$$W_i = \begin{cases} \lfloor \frac{i}{2} \rfloor \times \delta + K_{(\lfloor \frac{i}{2} \rfloor \times \delta)\&3} & \text{if } i \text{ is odd} \\ \lfloor \frac{i}{2} \rfloor \times \delta + K_{(\lfloor \frac{i}{2} \rfloor \times \delta \gg 11)\&3} & \text{if } i \text{ is even} \end{cases} \tag{5}$$

Here, $\delta = \lfloor (\sqrt{5} - 1) \times 2^{31}) \rfloor =$ `0x9e3779b9`. The subscripts to $K$ in the expression above simply define an expansion of $K_0, \ldots K_3$ words over the XTEA rounds. The expression $\lfloor \frac{i}{2} \rfloor \times \delta$ specifies round constants.

The round function is specified next in terms of Fig. 2 (b). Denote the 64-bit plaintext in the form of two 32-bit words $(A_0, A_1)$. Then, the encryption is done by calculating

$$r_{i+1} = r_{i-1} + (L(r_i) + r_i) \oplus W_i \tag{6}$$

for $i = 1, \ldots 64$, where $L(x) = (x \ll 4) \oplus (x \gg 5)$. The ciphertext is taken to be $(r_{63}, r_{64})$.

In [31], a related-key rectangle attack aiming to break 36 rounds of XTEA (rounds 16-51) and not requiring any weak-key assumptions is provided. The starting point for each rectangle attack is a family of top and bottom differential trails [4]. In [31], a family of trails is provided for $E_0$ (rounds 16-37) and one constant trail with probability 1 is provided for the bottom family (rounds 37-45). Then, each of the $E_0$ trails are connected to the fixed $E_1$ trail.

We used the URSA system to verify that the bottom trail cannot be connected to any of the trails in the top trail family. A particular high-probability representative pair of top-bottom trails (Table 3 in [31]) is shown in Table 2. The step numbers are given in the first and the last column, along with the active message words. Only the steps around the middle of the primitive are shown. Steps 35-37, where the contradiction can be localized, are marked in gray.

| step | $\Delta[A,B]=\Delta[C,D]$ | $\Delta[A,C]=\Delta[B,D]$ | step |
|---|---|---|---|
| | ⋮ | ⋮ | |
| 30 | `--------------------------------` | `????????????????????????????????` | 30 |
| 31 | `--------------------------------` | `????????????????????????????????` | 31 |
| 32 | `x-------------------------------` | `????????????????????????????????` | 32 |
| 33 | `-----x--------------------------` | `????????????????????????????????` | 33 |
| 34 | `xx---x----x---------------------` | `????????????????????????????????` | 34 |
| 35 | `-----x---------x----------------` | `????????????????????????????????` | 35 |
| 36 | `----------x---x----x-----------` | `x-------------------------------` | 36 |
| 37 | `-----x-x-----------------x------` | `--------------------------------` | 37 |
| 38 | `????????????????????????????????` | `--------------------------------` | 38 |
| 39 | `????????????????????????????????` | `--------------------------------` | 39 |
| | ⋮ | ⋮ | |

Table 2: One of the XTEA rectangle trails [31]

The bit-constraints provided by the top and the bottom trail in Table 2 are not fully propagated. Based only on the constraints given in the bottom trail in steps 36 and 37 and (6) for $i = 36$, one can conclude that $\Delta r_{35}^i[A,C] = \Delta r_{35}^i[B,D] =$ `'-'` for $i = 0,\ldots,25$ and $\Delta r_{35}^{26}[A,C] = \Delta r_{35}^{26}[B,D] =$ `'x'`. Taking into account these propagations, a detailed view of the relevant trail portion [31] is provided in Table 3.

In the proof below, let $C_\omega^i$ denote a carry bit at position $0 \le i \le 31$ on branch $\omega \in \{A,B,C,D\}$ in $r_{35}+(r_{36}+L(r_{36}))\oplus W_{36}$. We recall that in a 32-bit modular addition $z = x+y$, for $0 \le i \le 30$

$$z^{i+1} = x^{i+1} \oplus y^{i+1} \oplus c^i, \text{ where } c^i = maj(x^i, y^i, c^{i-1}) \tag{7}$$

while $c^{-1} = 0$.

| step | $\Delta[A,B]=\Delta[C,D]$ | $\Delta[A,C]=\Delta[B,D]$ |
|---|---|---|
| $\Delta r_{35}$ | `-----x---------x----------------` | `?????x----------------------------` |
| $\Delta r_{36}$ | `----------x---x----x-----------` | `x-------------------------------` |
| $\Delta L(r_{36})$ | `-------x---x--------x----x------` | `-----x----------------------------` |
| $\Delta s(r_{36}) = \Delta(r_{36}+L(r_{36}))\oplus W_{36})$ | `??????????????????????????x------` | `?????x----------------------------` |
| $\Delta r_{37} = \Delta(r_{35}+s(r_{36}))$ | `-----x-x-----------------x------` | `--------------------------------` |

Table 3: A detailed view of (contradictory) steps 35-37

**Observation 1** *Constraints specified in Table 3 are contradictory.*

*Proof:* The argument about the contradiction is split in two cases:

(i) Let the bit $s^{26}(r_{36})$ for both $\Delta[A,B]$ and $\Delta[C,D]$ be inactive. In Table 3, this bit constraint is shown in light gray and the assumption of this part of the proof replaces the `'?'` at this position by a `'-'`. As a consequence, $\Delta s^{27}(r_{36}) =$`'-'` in the $\Delta[A,B] = \Delta[C,D]$ column of Table 3, since $r_{36}$, $L(r_{36})$ and $W_{36}$ are inactive past bit-position 26.

It can be observed that $C_A^{25} = C_B^{25} = C_C^{25} = C_D^{25}$ and this carry value will be denoted by $C$. Namely, $C_A^{25} = C_C^{25}$ and $C_B^{25} = C_D^{25}$ since $\Delta s^i(r_{36}) = \Delta r_{35}^i =$`'-'` for $0 \le i \le 25$ in the $\Delta[A,C] = \Delta[B,D]$ column. Furthermore, in the $\Delta[A,B] = \Delta[C,D]$ column, $\Delta s^{26}(r_{36}) =$`'-'` due to the assumption and $\Delta r_{35}^{26} = \Delta r_{37}^{26} =$`'x'`. Thus, there is no carry disturbance from bit-position $i \le 25$ and $C_A^{25} = C_B^{25}$.

We show that both in the case $C = 0$ and the case $C = 1$, a contradiction is reached. According to the assumption of this part of the proof, the bit-value $s^{26}(r_{36})$ is equal to some fixed $b \in \{0, 1\}$ in both $A$ and $B$ branches. If $C = 0$, then $b = 0$ is a necessary condition, since if $b = 1$, the $\Delta r_{37}^{27}$ constraint would be `'x'` and this is not the case. However, since the $\Delta s^{26}(r_{36})$ is specified as `'x'` for $\Delta[A, C]$ and $\Delta[B, D]$, the necessary condition $b = 0$ cannot be fulfilled in $\Delta[C, D]$ and therefore this path cannot behave according to the $\Delta[A, B] = \Delta[C, D]$ column of Table 3. In the case $C = 1$, a necessary condition $b = 1$ is derived and the contradiction argument proceeds analogously.

(ii) Let the negation of the assumption used in (i) hold. In other words, let $s^{26}(r_{36})$ be active in $\Delta[A, B]$ or, let the same bit be active in $\Delta[C, D]$. This disjunction implies that both bits are active simultaneously, since $s^{26}(r_{36})$ is active in both $\Delta[A, C]$ and $\Delta[B, D]$. Next, we have that $s^{25}(r_{36})$ is active in both $\Delta[A, B]$ and $\Delta[C, D]$, since otherwise there would not be a carry difference coming from bit position 25 and causing the two active bits to sum to an active bit in $r_{37}^{26}$. Finally, it follows that bits $r_{37}^{27}$ in $\Delta[A, B]$ and $\Delta[C, D]$ are also active. This is true since $C_A^{26} \neq C_B^{26}$ and $C_C^{26} \neq C_D^{26}$. The first of the two equalities is true since both input bits and the output bit at position 26 are active when $r_{37}$ is calculated. Independently, the second inequality is valid for the same reason on $\Delta[C, D]$.

The `'x'` constraints on bit positions 25 and 26 in $\Delta s(r_{36})$ have the same sign (both `'u'` or both `'n'`), since they are caused by the carry propagation from bit position $i \leq 24$ in the $r_{36}$ and $L(r_{36})$ summation. This is true both in $\Delta[A, B]$ and $\Delta[C, D]$. On the other hand, this cannot hold, since the constraint at bit position 26 in $\Delta s(r_{36})$ at $[A, C]$ corrupts this sign and thus we have a contradiction. □

As already mentioned, we verified that the other top-bottom trail variants [31] are incompatible. It should be noted that all of the trails are induced by a difference at the most significant bit (MSB) positions in the key words. Previously, it was speculated [46] that if the top and the bottom trails start from the same bit position, contradictions are more likely to occur as the trails are likely to involve the same bit-positions. Our analysis confirms this intuition.

In this regard, one can also ask whether there exist *any* pair of compatible trails such that both top and the bottom trail are due to MSB disturbances in the round span discussed in [31] (31-37). Using URSA, this question can be answered by simply removing all of the trail constraints from the constraint representation and leave only those that enforce the top and the bottom trail expanded key disturbances. It should be noted that the task given to the SAT solver in this case is more difficult, since the solver has to effectively search for valid compatible differential trails. Increasing the number of rounds in the middle may result in impractical SAT solver execution times.

The following discussion is relevant at this point. To provide a lower bound for the probability of the distinguishing property used in the attack, most of the trails used in the previous literature on rectangle or boomerang attacks are *aligned* in the sense that the trails enforced on the opposite faces of the quartet structure the share the same active bit positions. This allows having only two trails to model all four faces in the quartet of primitive execution. However, previously, *unaligned* trails have also been attributed to add to the overall attack probability [4]. In such a case, the primitive follows four different trails and results in the desired output difference.

We verified whether there exist both aligned and unaligned solutions to the round span

discussed in [31]. The SAT solving phase for an aligned solution above took less than 30 minutes running as one process on 8-core 2.67 Ghz Intel i7 CPU before returning a negative answer. In other words, there exists no trails starting from the MSB positions in the 31-37 round span. However, interestingly enough, if the alignment constraints are removed, the solution does exist. The solution returned by the SAT solver follows four different (unaligned) trails and, as such, is different from the trails studied in the majority of previous literature (for rectangle attacks on block ciphers, see, e.g., [11, 32, 49] and as for boomerang distinguishers on hash functions see, e.g., [7, 8]). As we are not aware of previous examples of unaligned trails in the literature, the extracted trails are presented in Fig. 7 in the Appendix, along with the corresponding plaintext and key values in Fig. 8.

The analysis above shows that contradictions that occur because both top and bottom trails start from the most significant bit may be resolved if one allows unaligned trails. This is relevant in the context of building compression function distinguishers, since having boomerang trails induced by MSB disturbances reduces the complexity of the final phase of the second order collision search [7, 46].

## 3.2 On the incompatibility of SHACAL-1 trails [49, 12]

In 2001, Handschuh and Naccache [16, 17] proposed the SHACAL-1 block cipher and submitted it to the NESSIE (New European Schemes for Signatures, Integrity and Encryption) project [1]. SHACAL-1 is in fact the internal block cipher used within the SHA-1 hash function [43]. When applied in the Davies-Meyer mode, SHACAL-1 represents the SHA-1 compression function. Reduced-step SHACAL-1 was scrutinized both in the single-key and the related-key cryptanalytic models [5, 18, 27, 33]. As for the full-round SHACAL-1, it was shown to be susceptible to a rectangle related-key attack with complexity better than exhaustive search in [12] in 2006.

However, Wang *et al.* [49] found multiple problems in previous attacks on SHACAL-1. In particular, it was observed that the previous attacks [5, 18, 27, 33] do not work due to flaws in the provided differential trails. The trails turn out to be contradictory when regarded as single trails, i.e., independently of the quartet/rectangle context. Problems in these attacks are mostly related to the sign of active bits. In case only XOR differences are considered, these types of problems remain unnoticed [49].

Apart from finding flaws in previous attacks, [49] finds that the related-key rectangle attack [12] remains valid although it works against only a subset of the key space ($2^{496}$ out of $2^{512}$ keys). In addition, [49] proposed a new related-key rectangle attack that works for $2^{504}$ out of $2^{512}$ keys. To the best of our knowledge, these are the best attacks against SHACAL-1.

In this section, we show that the two attacks above are in fact also flawed. Although the trails are non-contradictory when regarded independently, once connected as specified by the rectangle setting, incompatible constraints are placed on the inner state bits. Moreover, below, we point out a particular type of contradiction that is likely to occur in rectangle attacks on ciphers with linear key schedule with good diffusion such as SHACAL-1. To the best of our knowledge, this type of rectangle/boomerang attack contradiction has not been discussed in the previous literature.

Below, a specification of the SHACAL-1 encryption function based on recurrence relations is provided. To encrypt, the 160-bit plaintext and the 512-bit key are copied to $(r_0, r_{-1}, r_{-2}, r_{-3}, r_{-4})$ and $(W_0, W_1, \ldots W_{15})$, respectively. The block cipher key is expanded according to the SHA-1

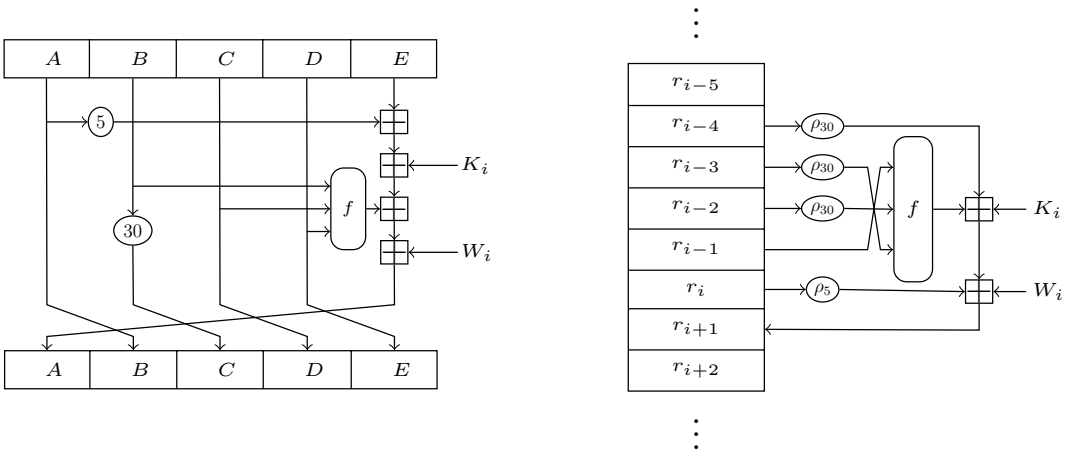


Figure 3: Two equivalent representations of the SHA-1 state update step

message expansion

$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1$$

for $i = 16, \ldots 79$. Next, 80 iterations of the function schematically represented in Fig. 3 are applied. Explicitly, for $i = 0, \ldots 79$, we have

$$r_{i+1} = r_{i-4} \lll \rho_{30}^i + K_i + f_i(r_{i-1}, r_{i-2} \lll \rho_{30}^i, r_{i-3} \lll \rho_{30}^i) + W_i + r_i \lll \rho_5^i$$

where $K_i$ are the round constants, $\rho_{30}^i = 30$ and $\rho_5^i = 5$ for $4 \leq i \leq 79$ and for $0 \leq i \leq 3$, the rotational constants are properly adjusted. The bit-wise logical functions are defined as:

$$f(x,y,z) = \begin{cases} IF(x,y,z) = (x \wedge y) \vee (\neg x \wedge z) & 0 \leq i \leq 19 \\ XOR(x,y,z) = x \oplus y \oplus z & 20 \leq i \leq 39 \text{ or } 60 \leq i \leq 79 \\ MAJ(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) & 40 \leq i \leq 59 \end{cases}$$

The SHACAL-1 ciphertext is defined to be $(r_{80}, r_{79}, r_{78}, r_{77}, r_{76})$.

In Table 4, contradictory portions of the SHACAL-1 trails are given (extracted from Tables 7 and 8 in [49]).

| step | $\Delta[A,B] = \Delta[C,D]$ | $\Delta W[A,B] = \Delta W[C,D]$ | $\Delta[A,C] = \Delta[B,D]$ | $\Delta W[A,C] = \Delta W[B,D]$ |
|---|---|---|---|---|
| 29 | ----------x--- | ---------- | ----------x- | xx--------x---xx |
| 30 | ----------x--- | ----------x | ----------xx | -x--------xx---x- |
| 31 | ----------x | ---------- | ----------- | x---------------x |
| 32 | ----------x---x- | ---------- | ----------x- | -x--------x---x- |
| 33 | ---x----------x---------x | ----------x- | ---x----------x- | -x----------x---x- |
| 34 | ????????????????????????????????? | ---------- | ----------- | -x---------------x |

Table 4: Incompatible SHACAL-1 trails [49]

**Observation 2** *Constraints specified in Table 4 are contradictory.*

*Proof:* As shown by gray bits in the third column of Table 4, only one input bit to $f_{33}$ for $[A, C]$ is active. Since $f_{33}$ is in fact the XOR function, the output $f_{33}$ bit at this position is

active as well. The $\Delta W_{33}^1[A, C] =$ `'x'` constraint cancels out this active bit since no bits are active in $\Delta r_{34}[A, C]$. This is possible only if the corresponding $f_{33}$ output bit and $\Delta W_{33}^1[A, C]$ have opposite signs. The same should hold for $[B, D]$ and this yields a contradiction since $\Delta W_{33}^1[A, B] = \Delta W_{33}^1[C, D] =$ `'x'`. □

As for the rectangle trails used in [12], we analyze the constraints in steps 57-63 in detail and show that these steps contain a contradiction. It should be noted that the top trail and the bottom trail for this attack cover steps 0-34 and 34-69, respectively. The contradictions are likely to occur in the region where both top and the bottom trails are specified, i.e., where the bottom and the top trails meet [42, 30, 46]. However, in this case, due to the message expansion linearity, the contradiction occurs in the late steps of the bottom trail as well.

In Table 5, trails for steps 57-63 are presented (Tables 2 and 3 in [12]). As can be observed, the $\Delta[A, B] = \Delta[C, D]$ column contains only `'?'` constraints since the top trail in late steps 57-63 is unspecified, as expected in the rectangle attack setting. However, since the message expansion in SHACAL-1 is linear, the $\Delta W[A, B] = \Delta W[C, D]$ is fully specified by the message expansion. The observation below shows that the linearly expanded constraints in the $\Delta W[A, B] = \Delta W[C, D]$ column do not allow the $\Delta[A, C] = \Delta[B, D]$ column to be satisfied.

| step | $\Delta[A,B]=\Delta[C,D]$ | $\Delta W[A,B]=\Delta W[C,D]$ | $\Delta[A,C]=\Delta[B,D]$ | $\Delta W[A,C]=\Delta W[B,D]$ |
|---|---|---|---|---|
| 57 | ???????????????????????????????????? | ----------x-----x--xxx--x-- | ---------- | ---------- |
| 58 | ???????????????????????????????????? | ----------x-xxx---xx-x-- | ---------- | ---------- |
| 59 | ???????????????????????????????????? | ----------x-xx-x---x--xx-- | ---------- | ---------- |
| 60 | ???????????????????????????????????? | ----------x-----xx--xxxx-- | ---------- | ---------- |
| 61 | ???????????????????????????????????? | ----------xx--xx-x-xxxx-- | ---------- | ----------x-- |
| 62 | ???????????????????????????????????? | ----------x-xxxx--xx-x-- | ----------x-- | ----------x------ |
| 63 | ???????????????????????????????????? | ----------x---x--xxx-x-xxxx-- | ---------- | ----------x-- |

Table 5: Incompatible SHACAL-1 trails [12]

**Observation 3** *Constraints specified in Table 5 are contradictory.*

*Proof:* According to Table 5, $\Delta W_{61}^2[A, C] =$ `'x'`. The sign of this constraint is equal to that of $\Delta r_{62}^2[A, C] =$ `'x'`, since all other input bits in the step 62 modular addition are inactive. The sign of $\Delta r_{62}^2[A, C] =$ `'x'` is opposite to the sign of $\Delta W_{62}^7[A, C]$ since these two constraints cancel out in step 63. Therefore, the sign of $\Delta W_{62}^7[A, C]$ is opposite to the sign of $\Delta W_{61}^2[A, C]$. The same holds for the $[B, D]$ face of the quartet. This yields a contradiction since $\Delta W_{61}^2[A, B] = W_{61}^2[C, D] =$ `'x'` and $\Delta W_{62}^7[A, B] = W_{62}^7[C, D] =$ `'-'`. □

It follows that constraints in steps even outside the switch region should be carefully verified for primitives with linear message expansions, such as SHA-1, SHACAL-1 and SM3.

## 3.3 On the incompatibility of SM3 trails [2, 3]

The SM3 hash function [19] is a cryptographic hashing standard in China adopted for use within the Trusted Computing framework in 2007 by the Chinese National Cryptographic Administration Bureau. It was designed by Xiaoyun Wang *et al.* and its design resembles the design of SHA-2 but includes additional fortifying features such as feeding two message-derived words into each step, as opposed to only one in the case of SHA-2.

SM3 is a Merkle-Damgård construction that processes 512-bit input message blocks and returns a 256-bit hash value. Since the attacks that we analyze below are attacks on the

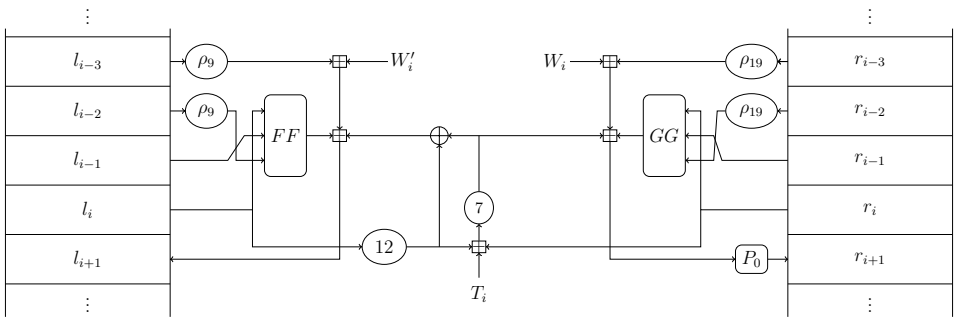


Figure 4: The SM3 state update step

compression function, the specification of compression function is provided below. For more details, the reader is referred to [19].

Let the $P_0$ and $P_1$ functions, both operating on 32-bit words, be defined by:

$$\begin{aligned} P_0(X) &= X \oplus (X \lll 9) \oplus (X \lll 17) \\ P_1(X) &= X \oplus (X \lll 15) \oplus (X \lll 23). \end{aligned}$$

The message block to be hashed is first represented as 16 32-bit words $M_0, \ldots, M_{15}$. Then, it is expanded to 68 32-bit words by letting $W_i = M_i$ for $0 \leq i < 16$ and

$$W_i = P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} \lll 15)) \oplus (W_{j-13} \lll 7) \oplus W_{j-6} \tag{8}$$

for $16 \leq i < 68$. We provide the specification of the step function using recurrence relations, similarly to the one used in [37]. The pre-fixed IV [19] is copied to $(l_0, l_{-1}, l_{-2}, l_{-3}, r_0, r_{-1}, r_{-2}, r_{-3})$ and the chaining values are computed over 64 steps as follows:

$$\begin{aligned} l_{i+1} &= FF_i(l_i, l_{i-1}, l_{t-2} \lll \rho_9) + l_{i-3} \lll \rho_9 + W_i \oplus W_{i+4} + SS1_i \oplus (l_i \lll 12) \\ r_{i+1} &= P_0(GG_i(r_i, r_{i-1}, r_{i-2} \lll \rho_{19}) + r_{i-3} \lll \rho_{19} + W_i + SS1_i) \end{aligned}$$

where $SS1_i = (l_i \lll 12 + r_i + T_i) \lll 7$. The functions $FF_i$ and $GG_i$ are defined by

$$FF_i(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z, & 0 \leq i \leq 15 \\ (X \wedge Y) \vee (Y \wedge Z) \vee (X \wedge Z) & 16 \leq i < 64 \end{cases}$$

$$GG_i(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z, & 0 \leq i \leq 15 \\ (X \wedge Y) \vee (\neg X \wedge Z) & 16 \leq i < 64 \end{cases}$$

The round constants are $T_i = 0x79cc4519 \lll i$ for $i \in \{0, \ldots, 15\}$ and $T_i = 0x7a879d8a \lll i$, for $i \in \{16, \ldots, 63\}$. As for the rotation constants, $\rho_9^i = 9$ and $\rho_{19}^i = 19$ for $2 \leq i \leq 63$ and for $0 \leq i < 2$, the rotational constants are properly adjusted.

Previous analysis of the reduced-step SM3 hash function includes preimage attacks [55, 50], collision attacks [37] and boomerang distinguishing attacks [28, 2, 3]. To the best of our knowledge, the highest number of steps is reached in [2], where an example of a boomerang quartet is provided for the 35-step reduced SM3 and attacks against 36, 37 and 38 step-reduced SM3 with complexities $2^{73.4}$, $2^{94}$ and $2^{192}$ are provided.

| step | $\Delta W'[A,B] = \Delta W'[C,D]$ | $\Delta W[A,B] = \Delta W[C,D]$ | $\Delta W'[A,C] = \Delta W'[B,D]$ | $\Delta W[A,C] = \Delta W[B,D]$ |
|---|---|---|---|---|
| 15 | x--------x-------x---------- | ---------------------------- | ---------------------------- | ---------------------------- |
| 16 | ---------------------------- | ---------------------------- | -x-xx---x-x-x-----xxx-------x--- | --------x--------------------- |
| 17 | ---------------------------- | ---------------------------- | ----------x----------x-x--- | ----------x----------x-x--- |
| 18 | ---x------------x-x---------- | ---------------------------- | ---------------------------- | ---------------------------- |
| 19 | x--------x-------x---------- | x--------x-------x---------- | --------x------------------- | ---------------------------- |
| | $\Delta l[A,B] = \Delta l[C,D]$ | $\Delta r[A,B] = \Delta r[C,D]$ | $\Delta l[A,C] = \Delta l[B,D]$ | $\Delta r[A,C] = \Delta r[B,D]$ |
| 15 | ---------------------------- | ---------------------------- | ---------------------------- | ---------------------------- |
| 16 | x--------x-------x----------x- | ---------------------------- | ----------------xx------x---x- | ---x-x-----xx-x----x-------xx--- |
| 17 | -----x-------x------x-x------xx- | ----xx-------x-------xx-----x-x- | ------xx---xx-x----x-x-----xx--- | ------x--------x-x--x--x-----x-x |
| 18 | ---x--x-x--x--xx-----x-x-xx----- | -xxx-xx---xxx--xxx--xx-xxxx--xx- | ----------------x----------- | ---------------------------- |
| 19 | ?????????????????????????????????? | ?????????????????????????????????? | ---------------------------- | ---------------------------- |

Table 6: Incompatible SM3 boomerang trails [3]

Below, we show that the 37 and 38-step distinguishers [3] are based on incompatible differentials. In Table 6, the incompatible portion of the trails is presented (based on Tables 6 and 7 in [3]). The fact that the message expansion in SM3 is linear allows extracting all the message bit-constraints. In the top part of the table, the message constraints both for $W'_i = W_i \oplus W_{i+4}$ and $W_i$ for $i = 15, \ldots 19$ are provided and in the bottom part the chaining values constraints are given. The bits relevant for the analysis are shaded in gray.

**Observation 4** *Constraints specified in Table 6 are contradictory.*

*Proof:* Recall that

$$l_{19} = FF_{18}(l_{18}, l_{17}, l_{16} \lll 9) + l_{15} \lll 9 + W'_{18} + SS1_{18} \oplus (l_{18} \lll 12) \tag{9}$$

where $SS1_{18} = (l_{18} \lll 12 + r_{18} + T_{18}) \lll 7$. Since according to Table 6, $\Delta W'_{18}[A,C]$, $\Delta l_{18}[A,C]$, $\Delta r_{18}[A,C]$ and $\Delta l_{15}[A,C]$ contain no active bits and the same is true for $\Delta l_{19}[A,C]$, we have that $\Delta FF_{18}(l_{18}, l_{17}, l_{16} \lll 9)[A,C]$ cannot have any active bits either. The same statement holds for $\Delta FF_{18}(l_{18}, l_{17}, l_{16} \lll 9)[B,D]$.

Consider the $FF_{18}$ input bits for bit-position 10 in the modular addition (9). The $FF_{18}$ input bit-constraints participating at this position are shaded in gray in Table 6. As can be observed, one of the input bits is active and, as established above, the function output bit is inactive. Since $FF_{18}$ is the majority logical function MAJ, it follows that $l^{10}_{18} = l^{1}_{16}$ in both branches $A$ and $C$. Again, the same statement holds for branches $B$ and $D$. However, this is impossible since $\Delta l^{10}_{18}[A,B] = l^{10}_{18}[C,D] =$ `'x'` and this is the only active input bit to $FF_{18}$ at branches $[A,B]$ and $[C,D]$. This shows that the constraints are incompatible. □

It is interesting to note that adding more freedom to the constraint set by removing the $\Delta l_{15}[A,C]$ and $\Delta l_{15}[B,D]$ constraints does not remove the contradiction.

# 4 Conclusion

The analysis provided in this paper shows that constructing rectangle or boomerang attacks should always be accompanied by formal verification of trails, since otherwise, there is little assurance that the trails are in fact compatible. Formal verification of trails should be performed whenever it is not possible to execute the attack in practice. An easy to use verification approach based on the URSA system was proposed.

Based on our analysis, the previous rectangle and boomerang attacks reaching the highest number of rounds against XTEA, SHACAL-1 and SM3 are shown to be based on incompatible differential trails. In addition, we pointed out a type of contradiction that is likely to occur in

primitives with fast-diffusion linear message expansions such as SHA/SHACAL-1. This type of contradictions have not been emphasized in previous literature. Finally, in the context of the XTEA block cipher, we provided examples of unaligned boomerang trails that contribute to the overall rectangle attack probability and are relevant in the area of boomerang distinguishers on hash functions.

# References


[1] NESSIE - New European Schemes for Signatures, Integrity and Encryption. `https://www.cosic.esat.kuleuven.be/nessie/`.

[2] Bai, D., Yu, H., Wang, G., and Wang, X. Improved Boomerang Attacks on Round-Reduced SM3 and BLAKE-256. Cryptology ePrint Archive, Report 2013/852, `http://eprint.iacr.org/`.

[3] Bai, D., Yu, H., Wang, G., and Wang, X. Improved Boomerang Attacks on SM3. In *ACISP* (2013), C. Boyd and L. Simpson, Eds., vol. 7959 of *Lecture Notes in Computer Science*, Springer, pp. 251–266.

[4] Biham, E., Dunkelman, O., and Keller, N. The Rectangle Attack - Rectangling the Serpent. In *EUROCRYPT* (2001), B. Pfitzmann, Ed., vol. 2045 of *Lecture Notes in Computer Science*, Springer, pp. 340–357.

[5] Biham, E., Dunkelman, O., and Keller, N. Rectangle attacks on 49-round SHACAL-1. In *FSE* (2003), pp. 22–35.

[6] Biham, E., and Shamir, A. Differential cryptanalysis of DES-like cryptosystems. In *CRYPTO* (1990), A. Menezes and S. A. Vanstone, Eds., vol. 537 of *Lecture Notes in Computer Science*, Springer, pp. 2–21.

[7] Biryukov, A., Lamberger, M., Mendel, F., and Nikolić, I. Second-order differential collisions for reduced SHA-256. In *ASIACRYPT* (2011), D. H. Lee and X. Wang, Eds., vol. 7073 of *Lecture Notes in Computer Science*, Springer, pp. 270–287.

[8] Biryukov, A., Nikolić, I., and Roy, A. Boomerang Attacks on BLAKE-32. In *FSE* (2011), A. Joux, Ed., vol. 6733 of *Lecture Notes in Computer Science*, Springer, pp. 218–237.

[9] Cannière, C. D., and Rechberger, C. Finding SHA-1 characteristics: General results and applications. In *ASIACRYPT* (2006), X. Lai and K. Chen, Eds., vol. 4284 of *Lecture Notes in Computer Science*, Springer, pp. 1–20.

[10] Chen, J., and Jia, K. Improved related-key boomerang attacks on round-reduced Threefish-512. In *ISPEC* (2010), J. Kwak, R. H. Deng, Y. Won, and G. Wang, Eds., vol. 6047 of *Lecture Notes in Computer Science*, Springer, pp. 1–18.

[11] Dunkelman, O., Fleischmann, E., Gorski, M., and Lucks, S. Related-key rectangle attack of the full HAS-160 encryption mode. In *INDOCRYPT* (2009), B. K. Roy and N. Sendrier, Eds., vol. 5922 of *Lecture Notes in Computer Science*, Springer, pp. 157–168.

[12] Dunkelman, O., Keller, N., and Kim, J. Related-Key Rectangle Attack on the Full SHACAL-1. In *Selected Areas in Cryptography* (2006), E. Biham and A. M. Youssef, Eds., vol. 4356 of *Lecture Notes in Computer Science*, Springer, pp. 28–44.

[13] Eibach, T., Pilz, E., and Völkel, G. Attacking Bivium using SAT solvers. In *Theory and Applications of Satisfiability Testing–SAT 2008*. Springer, 2008, pp. 63–76.

[14] Ganesh, V., Govostes, R., Phang, K., Soos, M., and Schwartz, E. STP - A Simple Theorem Prover (2006-2013). `http://stp.githubio/stp`.

[15] Gong, G., and Golomb, S. W. Transform domain analysis of DES. *IEEE Transactions on Information Theory 45*, 6 (1999), 2065–2073.

[16] Handschuh, H., Knudsen, L. R., and Robshaw, M. J. B. Analysis of SHA-1 in encryption mode. In *CT-RSA* (2001), D. Naccache, Ed., vol. 2020 of *Lecture Notes in Computer Science*, Springer, pp. 70–83.

[17] Handschuh, H., and Naccache, D. SHACAL. NESSIE, 2001.

[18] Hong, S., Kim, J., Lee, S., and Preneel, B. Related-key rectangle attacks on reduced versions of SHACAL-1 and AES-192. In *FSE* (2005), pp. 368–383.

[19] Internet Engineering Task Force. RFC: SM3 hash function, October, 2011. `tools.ietf.org/html/shen-sm3-hash-00`.

[20] Janičić, P. Uniform reduction to SAT. *Logical Methods in Computer Science 8*, 3 (2010).

[21] Jovanović, D., and Janičić, P. Logical analysis of hash functions. In *Frontiers of combining systems*. Springer, 2005, pp. 200–215.

[22] Jovanovic, P., Neves, S., and Aumasson, J.-P. Analysis of NORX. *IACR Cryptology ePrint Archive 2014* (2014), 317.

[23] Kamal, A. A., and Youssef, A. M. Applications of sat solvers to AES key recovery from decayed key schedule images. In *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on* (2010), IEEE, pp. 216–220.

[24] Kelsey, J., Kohno, T., and Schneier, B. Amplified boomerang attacks against reduced-round MARS and Serpent. In *FSE* (2000), B. Schneier, Ed., vol. 1978 of *Lecture Notes in Computer Science*, Springer, pp. 75–93.

[25] Khovratovich, D. Methods of Symmetric Key Cryptanalysis, 2011, available online: `http://research.microsoft.com/pubs/151070/state.pdf` .

[26] Kim, J., Hong, S., Preneel, B., Biham, E., Dunkelman, O., and Keller, N. Related-key boomerang and rectangle attacks: theory and experimental analysis. *Information Theory, IEEE Transactions on 58*, 7 (2012), 4948–4966.

[27] Kim, J., Kim, G., Hong, S., Lee, S., and Hong, D. The related-key rectangle attack - application to SHACAL-1. In *ACISP* (2004), H. Wang, J. Pieprzyk, and V. Varadharajan, Eds., vol. 3108 of *Lecture Notes in Computer Science*, Springer, pp. 123–136.

[28] Kircanski, A., Shen, Y., Wang, G., and Youssef, A. M. Boomerang and slide-rotational analysis of the SM3 hash function. In *Selected Areas in Cryptography* (2012), L. R. Knudsen and H. Wu, Eds., vol. 7707 of *Lecture Notes in Computer Science*, Springer, pp. 304–320.

[29] Knudsen, L. R., and Robshaw, M. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011.

[30] Leurent, G. Analysis of differential attacks in ARX constructions. In *ASIACRYPT* (2012), X. Wang and K. Sako, Eds., vol. 7658 of *Lecture Notes in Computer Science*, Springer, pp. 226–243.

[31] Lu, J. Related-key rectangle attack on 36 rounds of the XTEA block cipher. *Int. J. Inf. Sec. 8*, 1 (2009), 1–11.

[32] Lu, J., and Kim, J. Attacking 44 Rounds of the SHACAL-2 Block Cipher Using Related-Key Rectangle Cryptanalysis. *IEICE Transactions 91-A*, 9 (2008), 2588–2596.

[33] Lu, J., Kim, J., Keller, N., and Dunkelman, O. Differential and rectangle attacks on reduced-round SHACAL-1. In *INDOCRYPT* (2006), R. Barua and T. Lange, Eds., vol. 4329 of *Lecture Notes in Computer Science*, Springer, pp. 17–31.

[34] Lu, J., Kim, J., Keller, N., and Dunkelman, O. Related-key rectangle attack on 42-round SHACAL-2. In *ISC* (2006), S. K. Katsikas, J. Lopez, M. Backes, S. Gritzalis, and B. Preneel, Eds., vol. 4176 of *Lecture Notes in Computer Science*, Springer, pp. 85–100.

[35] Massacci, F., and Marraro, L. Logical cryptanalysis as a SAT problem. *Journal of Automated Reasoning 24*, 1-2 (2000), 165–203.

[36] Mendel, F., Nad, T., and Schläffer, M. Finding SHA-2 characteristics: Searching through a minefield of contradictions. In *ASIACRYPT* (2011), D. H. Lee and X. Wang, Eds., vol. 7073 of *Lecture Notes in Computer Science*, Springer, pp. 288–307.

[37] Mendel, F., Nad, T., and Schläffer, M. Finding collisions for round-reduced SM3. In *CT-RSA* (2013), E. Dawson, Ed., vol. 7779 of *Lecture Notes in Computer Science*, Springer, pp. 174–188.

[38] Mendel, F., Nad, T., and Schläffer, M. Improving Local Collisions: New Attacks on Reduced SHA-256. In *EUROCRYPT* (2013), T. Johansson and P. Q. Nguyen, Eds., vol. 7881 of *Lecture Notes in Computer Science*, Springer, pp. 262–278.

[39] Mironov, I., and Zhang, L. Applications of SAT solvers to cryptanalysis of hash functions. In *SAT* (2006), A. Biere and C. P. Gomes, Eds., vol. 4121 of *Lecture Notes in Computer Science*, Springer, pp. 102–115.

[40] Morawiecki, P., and Srebrny, M. A SAT-based preimage analysis of reduced KECCAK hash functions. *Information Processing Letters 113*, 10 (2013), 392–397.

[41] Mouha, N., and Preneel, B. Towards finding optimal differential characteristics for arx: Application to Salsa20. `http://eprint.iacr.org/`.

[42] Murphy, S. The return of the cryptographic boomerang. *IEEE Transactions on Information Theory 57*, 4 (2011), 2517–2521.

[43] National Institute of Standards and Technology. USA, Secure Hash Standard FIPS 180-2. 2002.

[44] Prokop, L. Using SAT Solvers to Detect Contradictions in Differential Characteristics. Advisors: F. Mendel, M. Schläffer, April 2014, `http://lukas-prokop.at/proj/bakk_iaik/thesis.pdf`.

[45] R.-P., Weinmann. The ARX Challenge, Fast Software Encryption (FSE) 2009, Rump Session.

[46] Sasaki, Y. Boomerang distinguishers on md4-family: First practical results on full 5-pass haval. In *Selected Areas in Cryptography* (2011), A. Miri and S. Vaudenay, Eds., vol. 7118 of *Lecture Notes in Computer Science*, Springer, pp. 1–18.

[47] Soos, M., Nohl, K., and Castelluccia, C. Extending SAT solvers to cryptographic problems. In *Theory and Applications of Satisfiability Testing-SAT 2009*. Springer, 2009, pp. 244–257.

[48] Wagner, D. The boomerang attack. In *FSE* (1999), L. R. Knudsen, Ed., vol. 1636 of *Lecture Notes in Computer Science*, Springer, pp. 156–170.

[49] Wang, G., Keller, N., and Dunkelman, O. The Delicate Issues of Addition with Respect to XOR Differences. In *Selected Areas in Cryptography* (2007), pp. 212–231.

[50] Wang, G., and Shen, Y. Preimage and pseudo-collision attacks on step-reduced SM3 hash function. *Inf. Process. Lett. 113*, 8 (2013), 301–306.

[51] Wang, X., Yin, Y. L., and Yu, H. Finding collisions in the full SHA-1. In *CRYPTO* (2005), V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer, pp. 17–36.

[52] Wang, X., and Yu, H. How to break MD5 and other hash functions. In *EUROCRYPT* (2005), R. Cramer, Ed., vol. 3494 of *Lecture Notes in Computer Science*, Springer, pp. 19–35.

[53] Wheeler, D. J., and Needham, R. M. TEA Extensions. Technical Report, Computer Laboratory, University of Cambridge, 1997.

[54] Yun, A., Sung, S. H., Park, S., Chang, D., Hong, S., and Cho, H.-S. Finding collision on 45-step HAS-160. In *ICISC* (2005), D. Won and S. Kim, Eds., vol. 3935 of *Lecture Notes in Computer Science*, Springer, pp. 146–155.

[55] Zou, J., Wu, W., Wu, S., Su, B., and Dong, L. Preimage attacks on step-reduced SM3 hash function. In *ICISC* (2011), pp. 375–390.

The URSA file corresponding to the incompatible boomerang trail example in Section 2.2:

```
nzA = nxA + nyA;
nzB = nxB + nyB;
nzC = nxC + nyC;
nzD = nxD + nyD;

bxAB = (nxA == nxB);   /* trail (1)   ('----') */
bxCD = (nxC == nxD);   /* trail (1)   ('----') */
bxAC = (nxA == nxC);   /* trail (2)   ('----') */
bxBD = (nxB == nxD);   /* trail (2)   ('----') */

byAB = ( (nyA ^ nyB) == 1 );   /* trail (1) ('---x') */
byCD = ( (nyC ^ nyD) == 1 );   /* trail (1) ('---x') */
byAC = ( (nyA ^ nyC) == 1 );   /* trail (2) ('---x') */
byBD = ( (nyB ^ nyD) == 1 );   /* trail (2) ('---x') */

bzAB = ( (nzA ^ nzB) == 1);     /* trail (1) ('---x') */
bzCD = ( (nzC ^ nzD) == 1);     /* trail (1) ('---x') */
bzAC = ( (nzA ^ nzC) == 3);     /* trail (2) ('--xx') */
bzBD = ( (nzB ^ nzD) == 3);     /* trail (2) ('--xx') */

assert( bxAB && bxCD && bxAC && bxBD &&  byAB && byCD && byAC && byBD
&&   bzAB && bzCD && bzAC && bzBD );
```

| step | $\Delta[A,B]$ | $\Delta[C,D]$ | $\Delta[A,C]$ | $\Delta W[B,D]$ |
|---|---|---|---|---|
| 30 | ---------- | ---------- | nnun--u-----u-----nn-un---nun-n- | nnun--u-----u-----nn-un---nun-n- |
| 31 | ---------- | ---------- | -----nu--u-uu---nn---nunuu------ | -----nu--u-uu---nn---nunuu------ |
| 32 | u--------- | u--------- | -n-n-n----nnnnu--nnuu---------- | -n-n-n----nnnnu--nnuu---------- |
| 33 | -----n---- | -----n---- | n-u-u-u-nuu-u--n--------------- | n-u-u-u-nuu-u--n--------------- |
| 34 | -uu--uunnuu- | un-unn----n | nuu-uu-nnuu------------------- | -n-u-nn---n------------------- |
| 35 | -nu-nn--u-uu-nnn------------ | u---nu--u-uu-nnn------------ | -n-u-n------------------------ | u-nu-u------------------------ |
| 36 | u--un---u---nu-nnnu-n------- | n--un---u---nu-nnnu-n------- | u--------------------------- | n--------------------------- |
| 37 | --nnu------uun-nun-----unu----- | --nnu------uun-nun-----unu----- | ---------- | ---------- |
| 38 | -n-n----u-uun-u--u---unuu-nnu-n- | -n-n----u-uun-u--u---unuu-nnu-n- | ---------- | ---------- |

Table 7: XTEA boomerang trails with unaligned constraints (marked in gray)

| Key quartet | | | | |
|---|---|---|---|---|
| $K_A$ | 0x4c266470 | 0x616feff | 0x98254f67 | 0x6a6714 |
| $K_B$ | 0x4c266470 | 0x616feff | 0x98254f67 | 0x806a6714 |
| $K_C$ | 0x4c266470 | 0x616feff | 0x18254f67 | 0x6a6714 |
| $K_D$ | 0x4c266470 | 0x616feff | 0x18254f67 | 0x806a6714 |
| Plaintext quartet | | | | |
| $P_A$ | 0x259f4198 | 0xfb5ae217 | | |
| $P_B$ | 0x27ed0f0c | 0xe49bdb36 | | |
| $P_C$ | 0xe8422de | 0xfc22d87a | | |
| $P_D$ | 0x7fa55484 | 0x8b285daf | | |

Table 8: XTEA unaligned quartet example