Differential Cryptanalysis of Round-Reduced Sparx-64/128

Ralph Ankele[1]*, Eik List[2]

[1] Royal Holloway University of London, United Kingdom.
`ralph.ankele.2015@rhul.ac.uk`
[2] Bauhaus-Universität Weimar, Germany.
`eik.list@uni-weimar.de`

Abstract. SPARX is a family of ARX-based block ciphers designed according to the *long-trail strategy* (LTS) that were both introduced by Dinu et al. at ASIACRYPT'16. Similar to the wide-trail strategy, the LTS allows provable upper bounds on the length of differential characteristics and linear paths. Thus, the cipher is a highly interesting target for third-party cryptanalysis. However, the only third-party cryptanalysis on SPARX-64/128 to date was given by Abdelkhalek et al. at AFRICACRYPT'17 who proposed impossible-differential attacks on 15 and 16 (out of 24) rounds.
In this paper, we present chosen-ciphertext differential attacks on 16 rounds of SPARX-64/128. First, we show a truncated-differential analysis that requires 2^{32} chosen ciphertexts and approximately 2^{93} encryptions. Second, we illustrate the effectiveness of boomerangs on SPARX by a rectangle attack that requires approximately $2^{59.6}$ chosen ciphertexts and about $2^{122.2}$ encryption equivalents. Finally, we also considered a yoyo attack on 16 rounds that, however, requires the full codebook and approximately 2^{126} encryption equivalents.

Keywords: Symmetric-key cryptography · cryptanalysis · boomerang · truncated differential · yoyo · ARX.

1 Introduction

ARX CIPHERS. The design and cryptanalysis of block ciphers is a heuristic competition between designers and analysts. With the introduction of the wide-trail design strategy in Rijndael, designers could finally provide provable bounds for the expected probabilities and therefore for the maximal length of differential characteristics and linear trails of block ciphers. Rijndael and similar designs are substitution-permutation networks (SPNs), which left the earlier path of using only the omnipresent modular addition, XOR, rotation, and shift operations that

* This research was partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-MSCA-ITN-2014-643161 ECRYPT-NET.

nearly every processor supports out-of-the-box. As a consequence, SPNs demand an expertised tailoring of their implementations to the operating platform to be comparably efficient as bit-based designs. However, in resource-constrained environments, the most efficient software implementations are still ciphers that employ only logical operations and/or addition, e.g., ciphers based on modular additions, rotations, and XOR (ARX). Hence, until recently, there has been a gap between the provable bounds of wide-trail designs, and the efficiency of ARX-based constructions.

Sparx. At ASIACRYPT'16, Dinu et al. introduced Sparx [8], the first ARX-based family of block ciphers that provides provable bounds on the maximal length of differential characteristics and linear trails. Alongside Sparx, the authors developed the long-trail design strategy, a general approach for ARX-based symmetric-key primitives to obtain provable bounds. Both the long-trail strategy in general, and Sparx in particular, are interesting targets of cryptanalysis as they try to bridge the gap between efficiency and providing security bounds. The question arises if it is also secure against (truncated) differential and boomerang attacks that can exploit clustering effects of many differential characteristics.

Research Gap and Related Work. In the specification of Sparx, the designers reported on their results of a first automated analysis that no differential characteristic with probability higher than 2^{-n} nor any linear characteristic with bias higher than $2^{-n/2}$ exists over five or more steps. Moreover, they described integral attacks on up to five out of eight steps of Sparx-64/128, and six out of ten steps of Sparx-128. Though, those initial attacks are naturally limited due to time constraints when designing a new cipher, and therefore demand a deeper analysis by the cryptographic community. At AFRICACRYPT'17, Abdelkhalek et al. [1] proposed 12- and 13-round impossible-differential distinguishers on Sparx-64/128, using the four-step distinguisher for balanced Type-1 Feistel networks. They extended their attacks by three rounds, respectively, where they exploited dependencies between the key words from the key-schedule. Recently, Tolba et. al. proposed multi-dimensional zero-correlation linear attacks on up to 26 rounds of Sparx-128/128, and on up to 29 rounds of Sparx-128/256 [15].

Contribution and Outline. This work adds two chosen-ciphertext attacks on Sparx-64/128 in the single-key model: (1) a truncated-differential attack on 16 rounds and (2) a rectangle attack on 16 rounds; moreover, we further considered yoyo attacks on the same number of rounds in Appendix A, that, however, demands the full codebook. Table 1 compares their parameters with the previous attacks on Sparx-64/128 from the literature. In the remainder, we briefly revisit the necessary notions as well as the details of Sparx-64 in Section 2. We describe our truncated-differential attack in Section 5. We continue with an informal introduction to boomerang and rectangle attacks in Section 3. In Section 4, we describe our search of differential trails before we detail our rectangle attack on Sparx-64/128 in Section 6. Appendix A outlines a yoyo attack on 16 rounds. Finally, Section 7 concludes this work.

Table 1: Previous and proposed attacks on Sparx-64/128. KP/CP/CC = known plaintext/chosen plaintext/chosen ciphertext. ID = Impossible differential, TD = Truncated differentials.

Rounds	Attack type	Time	Data	Memory	Ref.
15/24	Integral	$2^{101.0}$	$2^{37.0}$ CP	$2^{64.0}$	[8]
15/24	ID	$2^{94.1}$	$2^{51.0}$ CP	$2^{43.5}$	[1]
16/24	ID	$2^{94.0}$	$2^{61.5}$ KP	$2^{61.5}$	[1]
16/24	TD	$2^{93.0}$	$2^{32.0}$ CC	$2^{61.0}$	Sect. 5
16/24	Rectangle	$2^{122.2}$	$2^{59.6}$ CC	$2^{61.6}$	Sect. 6
16/24	Yoyo	$2^{126.0}$	$2^{64.0}$ CP	$2^{64.0}$	App. A

2 Preliminaries

General Notations. We denote by $\mathbb{F}_2$ the finite field of two elements $x \in \{0,1\}$. For positive integer n, we denote by $\mathbb{F}_2^n$ the space of n-element vectors from $\mathbb{F}_2$. We represent functions by upper case letters and indices by lowercase letters. $\{0,1\}^n$ is the set of all n-bit strings and $\{0,1\}^*$ the set of bit strings of arbitrary length. Let $x, y \in \{0,1\}^n$ for some positive integer n in the following. Then, we denote by $x \,\|\, y$ the concatenation of x and y, by $x \oplus y$ their bitwise XOR, by $x \lll r$ a rotation by r bit to the left and by $x \ggg r$ rotation by r bit to the right; moreover, we denote by $x \boxplus y = (x+y) \bmod 2^n$ modular addition, and by $x \boxminus y = (x-y) \bmod 2^n$ modular subtraction. For all bit strings $x \in \{0,1\}^n$, we index the bits $x = (x_{n-1} \dots x_1 x_0)$ where x_{n-1} is the most significant and x_0 the least significant bit of x. Given a bit string $x = (x^1 \,\|\, \dots \,\|\, x^m) \in (\{0,1\}^{mn})$ consisting of m words of n bit each, we denote by

$$x \lll_n r \overset{\text{def}}{=} (x^1 \lll r) \,\|\, \dots \,\|\, (x^m \lll r)$$

the word-wise independent rotated value. We overload the notation for tuples of bit strings $x \in (\{0,1\}^n)^m$: $x = (x^1, \dots, x^m)$, to still mean wordwise independent rotation $x \lll_n r \overset{\text{def}}{=} (x^1 \lll r), \dots, (x^m \lll r)$. We use typewriter font to represent hexadecimal values, e.g., `0110` = 272. We use the same font but with annotation to represent bit strings, e.g., $(\texttt{0110})_2 = 6$; moreover, we will use the symbol `*` at the position of that certain bits to indicate that they can take arbitrary values, e.g., $(\texttt{0*10})_2 \in \{2, 6\}$. As a shorthand notation for probabilities p, we often write $h_w = -\log_2(p)$ when the meaning of p is clear from the context.

2.1 The Sparx Family of Ciphers

The Sparx-n/k family comprises three versions, Sparx-64/128, Sparx-128/128, and Sparx-128/256, where n indicates the block size, and k the key length k.

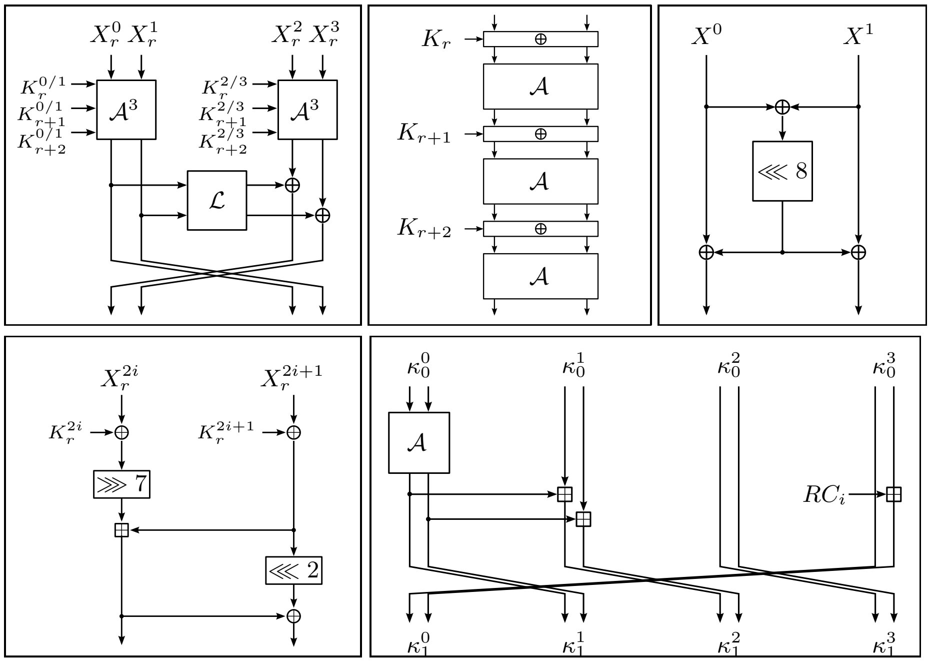


Fig. 1: High-level view of SPARX-64. **Top left:** The step function. **Top center:** The $\mathcal{A}^3$ layer in SPARX-64. **Top right:** The linear layer $\mathcal{L}$. **Bottom left:** The $\mathcal{A}$ function SPECKEY. **Bottom right:** One iteration of the key schedule of SPARX-64/128.

The cipher is based on a Feistel network with two state words for SPARX-64 and four state words for SPARX-128, consisting of n_s Feistel steps. Each step consists of r_a rounds of an ARX-based round function; plain- and ciphertexts consist of $w = n/32$ words $X^0, \ldots, X^{w-1}$ of 32 bit each; the key is divided into 32-bit words $(\kappa^0, \ldots, \kappa^{v-1})$. The values for the individual versions of SPARX are summarized in Table 2, the components of the cipher are also depicted in Figure 1.

SPARX-64/128. The structure of SPARX-64 is reminiscent of a Feistel network of eight steps. Each step consists of $r_a = 3$ rounds of the ARX-box $\mathcal{A}$, (i.e. three rounds of SPECKEY) on each branch. The Feistel function $\mathcal{L}$ is a linear involutive permutation $\mathcal{L} : \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ inspired by [9]. Given the left 32-bit state word $x \,\|\, y$, the input is split into 16-bit parts x, y, and is mapped to

$$\mathcal{L}(x \,\|\, y) \stackrel{\text{def}}{=} (x \oplus ((x \oplus y) \lll 8)) \,\|\, (y \oplus ((x \oplus y) \lll 8)).$$

We denote the 64-bit state after Round r interchangeably as $(L_r, R_r) = (X_r^0 \,\|\, X_r^1, X_r^2 \,\|\, X_r^3) = (X_r^L \,\|\, Y_r^L, X_r^R \,\|\, Y_r^R)$, and the round key used in Round r interchangeably as $(K_r^L,\, K_r^R) = (K_r^0 \,\|\, K_r^1,\, K_r^2 \,\|\, K_r^3)$.

THE KEY SCHEDULE OF SPARX-64. The 128-bit secret key of SPARX-64/128 is divided into four initial 32-bit words $(\kappa_0^0, \kappa_0^1, \kappa_0^2, \kappa_0^3)$. In each step, the key

Table 2: Parameters of the individual versions of Sparx.

Cipher	#State-words w	#Key-words v	#Rounds/step r_a	#Steps n_s
Sparx-64/128	2	2	3	8
Sparx-128/128	4	4	4	8
Sparx-128/256	4	8	4	10

schedule transforms the leftmost 32-bit word κ_s^0 in one iteration of the ARX-box $\mathcal{A}$, adds the output to the right neighboring word κ_s^1, adds a round constant RC_i to the rightmost 16-bit half of κ_{2s}^3 to prevent slide attacks, and finally rotates the four words by one position to the right. The $r_a = 3$ leftmost words κ_{2s}^0, κ_{2s}^1, κ_{2s}^2 are used as round keys for the first, second, and third round of the left branch of Step $s+1$; the $r_a = 3$ left-most words κ_{2s+1}^0, κ_{2s+1}^1, κ_{2s+1}^2 are used for the first, second, and third round of the right branch of Step $s+1$. For example, $(\kappa_0^0, \kappa_0^1, \kappa_0^2)$ are used as round keys for the left branch in the first step, and $(\kappa_1^0, \kappa_1^1, \kappa_1^2)$ are used as round keys for the right branch in the first step.

2.2 Properties

As observed by Abdelkhalek et al. [1], one can obtain the rounds keys for 2.5 consecutive rounds by guessing only 64 bit of key material. More precisely, one obtains the round keys for Round $3r+1$ and the round key for the right 32-bit branch in Round $3r+2$ by guessing the 64 bit of the key material of Round $3r$:

Property 1. Given κ_{s+1}^2 and κ_{s+1}^3, one can directly derive the key words $\kappa_s^2 = \kappa_{s+1}^3$, $\kappa_{s+2}^0 = \kappa_{s+1}^3$, $\kappa_{2s+3}^1 = \mathcal{A}(\kappa_{s+2}^0)$, and $\kappa_{s+3}^0 = \kappa_{s+1}^2$.

We learnt the following Property 2 from Leurent [11]. It reduces the effort of studying all combinations of pairs to that of comparing their outputs from F. One can further reduce the rank of F to $n-d$ so that outputs of F collide if and only if their inputs have one of 2^d differences.

Property 2. Assume, $\Delta \in \mathbb{F}_2^n$ is a fixed difference, and $x^0, \ldots, x^m \in \mathbb{F}_2^n$ represent m values for which the goal is to find pairs (x^i, x^j) that result in $x^i \oplus x^j = \Delta$. Then, one can define a linear function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ with rank $n-1$, s. t. $F(\Delta) = 0^n$; thus, all pairs (x^i, x^j) with $x^i \oplus x^j = \Delta$ will collide in $F(x^i) = F(x^j)$.

3 Boomerang and Rectangle Attacks

Boomerang attacks, as proposed by Wagner [17], allow an attacker to concatenate two short differentials with high probability when long differentials with sufficient probability are absent or hard to find. In the basic setting, an adversary decomposes an encryption function $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ into two subciphers $E = E_2 \circ E_1$, such that $E(P) \stackrel{\text{def}}{=} E_2(E_1(P))$. Then, it considers a first differential $\alpha \to \beta$ with probability p over E_1 and a second differential $\gamma \to \delta$ with probability q over E_2. These are often called upper and lower differentials or trails, respectively. They can then be combined in a chosen-plaintext and adaptive chosen-ciphertext attack to construct a boomerang distinguisher that consists of the following steps:

1. Choose a plaintext pair (P, P') with difference $\alpha = P \oplus P'$ and encrypt it through E to obtain its ciphertext pair (C, C') with difference β.
2. Derive $D = C \oplus \delta$ and $D' = C' \oplus \delta$ (the δ-shift) and decrypt D and D' through E^{-1} to obtain the corresponding plaintext pair (Q, Q').
3. If the plaintext pair (Q, Q') has difference $\alpha = Q \oplus Q'$, then (P, P', Q, Q') form a correct quartet.

Proposition 1. For a quartet (P, P', Q, Q'), there exists a differential with an input difference α for $P' = P \oplus \alpha$, $Q' = Q \oplus \alpha$, and a corresponding output difference β for $U' = U \oplus \beta$, $V' = V \oplus \beta$ with probability p. If we consider a differential $\delta \to \gamma$ with input difference $D = C \oplus \delta$, $D' = C' \oplus \delta$ and a corresponding output difference γ for $V = U \oplus \gamma$, it holds with probability q that $V' = U' \oplus \gamma$. Then, we can connect both differentials if we consider $V = U \oplus \gamma$, it follows that $V' = V \oplus \beta = (U \oplus \gamma) \oplus \beta = (U \oplus \beta) \oplus \gamma = U' \oplus \gamma$.

Calculating the probabilities for a correct quartet requires to consider both plaintext pairs (P, P') and (Q, Q') and results in a probability of $(pq)^2$. For the differentials to exist, the resulting probability has to satisfy $(pq)^2 \geq 2^{-n/2}$.

The probability of a correct quartet can be improved if one fixes input differences α and δ but allows all possible differences for β and γ, with the only requirement that $\beta \neq \gamma$. A boomerang distinguisher would then consider all trails of the form $\alpha \to \beta'$ for the upper trail and $\delta \to \gamma'$ for the lower trail. This increases the probability to $(\widehat{pq})^2$ where $\widehat{p} = \sqrt{\sum_{\beta'} \Pr^2\left[\alpha \to \beta'\right]}$ and $\widehat{q} = \sqrt{\sum_{\gamma'} \Pr^2\left[\delta \to \gamma'\right]}$ where $\widehat{p}$ is evaluated over E_1 and $\widehat{q}$ over E_2^{-1}, respectively.

The Rectangle Attack. In boomerang attacks, the adversary needs to query its oracles with chosen plaintexts and adaptively chosen ciphertexts. Since our boomerang attack will have to guess a considerable amount of key bits, which would require an oracle query for every obtained text and key guess, we will employ a *rectangle* attack instead. *Rectangle* attacks [3] have been derived from the *amplified boomerang* [10], both of which transform the boomerang into a purely chosen-plaintext attack (or chosen-ciphertext, if the adversary starts from the opposite direction). The core idea is to encrypt many pairs (P, P') with difference $P' \oplus P = \alpha$ in the hope that some of those will form a quartet with the desired differences in the middle with probability 2^{-n}. Given N plaintext pairs, the number of correct quartets is reduced to $N^2 \cdot 2^{-n} \cdot (\widehat{pq})^2$. Note that two pairs (U, U') and (V, V') can be combined in two distinct ways to a quartet in the middle: $U \oplus V = U' \oplus V' = \beta$ or $U \oplus U' = V \oplus V' = \beta$. [4] presented further improvements to the technique. The disadvantages of rectangle compared to boomerang attacks are the increased data complexity and the large number of potential quartets that have to be filtered to find correct quartets.

Table 3: An optimal six-round differential trail.

Rd.	ΔL_i	ΔR_i	h_w	
0	00000000	02110a04	–	–
1	00000000	28000010	0	4
2	00000000	00400000	0	2
3	00000000	80008000	0	0
$\mathcal{L}$	80008000	00000000	0	0
4	81008102	00000000	1	0
5	8000840a	00000000	2	0
6	850a9520	00000000	4	0
$\mathcal{L}$	af1abf30	850a9520	0	0

Table 4: Optimal differentials through up to ten rounds of Sparx-64; t is the run time of each search.

#Rds.	Δ_{in}	Δ_{out}	h_w	t
1	(00408000, 00000000)	(00000002, 00000000)	0.00	0.02s
2	(00102000, 00000000)	(80008002, 00000000)	1.00	0.10s
3	(28000010, 00000000)	(83008302, 81008102)	3.00	0.46s
4	(00000000, 28000010)	(8000840a, 00000000)	4.99	2.40s
5	(00000000, 02110a04)	(8000840a, 00000000)	8.99	25.07s
6	(00000000, 02110a04)	(af1abf30, 850a9520)	12.99	0.06h
7	(00000000, 14881008)	(82048e0e, 8000840a)	23.95	47.80h
8	(00000000, 540a0120)	(8000840a, 8000840a)	28.53	15.20d
9	(28000010, 28000010)	(d2609263, d1209123)	32.87	22.30d
10	(28000010, 28000010)	(80818283, 80008002)	38.12	32.50d

Ladder Switch. There exist a few approaches for increasing the transitional probability of boomerang trails in the middle, i.e., in the switching phase between top and bottom trial. Two well-established approaches are the Feistel switch and ladder switch; recently, Sasaki et al. [7] observed a number of more ways. Here, we concentrate on the ladder switch by [5]. It exploits that start and end of upper and lower trails can be located at different locations for each part of the state. For Sparx, it is intuitive to consider full steps: e.g., assume that the top trail has a nonzero difference in the left branch through the step in the middle. If the right branch has a zero difference in the left branch, then one can put the switch for the left branch before the step and consider it to be part of the bottom trail, which has probability one. Clearly, this approach can be generalized further to probabilities smaller than one. For Sparx, an optimal switch has one active (e.g., the left) and one inactive (e.g., the right) branch in the top trail, and mirrored in the bottom trail (e.g., right active and left inactive), which allows to pass the step in the switching phase with probability one.

4 Differential Trails and Boomerang Distinguishers

We employed a two-step approach: first, we searched for optimal differential characteristics for up to ten rounds of Sparx-64. Those formed the base of the wrapping rounds before and after the boomerang switches. Thereupon, we considered three interesting types of boomerangs over five steps.

4.1 Searching Optimal Differential Trails

We implemented variants of Sparx in CryptoSMT [14], an open-source tool based on the SAT/SMT solvers CryptoMiniSat [12] and STP [16] to search for

Table 5: Top (left to right): best trails found for our differentials of Type 1a, Type 1b, Type 1c, and Type 1d. **Middle**: best trails found for our differentials of Type 2a, Type 2b, Type 2d, and Type 2e. **Bottom**: Type 2c, Type 3a, Type 3b, and Type 3c. $\varSigma$ denotes the sum of h_w over all rounds.

Rd.	ΔL_i	ΔR_i	h_w
0	00000000	28000010	– –
1	00000000	00400000	0 2
2	00000000	80008000	0 0
3	00000000	81008102	0 1
$\mathcal{L}$	81008102	00000000	0 0
$\varSigma$			3

Rd.	ΔL_i	ΔR_i	h_w
0	28000010	28000010	– –
1	00400000	00400000	2 2
2	80008000	80008000	0 0
3	81008102	83008302	1 2
$\mathcal{L}$	00000000	81008102	0 0
$\varSigma$			7

Rd.	ΔL_i	ΔR_i	h_w
0	40404000	00400000	– –
1	40804081	80008000	2 0
2	40004205	81008102	3 1
3	42854a90	8000840a	5 2
$\mathcal{L}$	d78ddb92	42854a90	0 0
$\varSigma$			13

Rd.	ΔL_i	ΔR_i	h_w
0	80008000	80008000	– –
1	81008102	81008102	1 1
2	8004840e	8004840e	3 3
3	bd1aad20	870a9730	7 8
$\mathcal{L}$	00000000	bd1aad20	0 0
$\varSigma$			23

Rd.	ΔL_i	ΔR_i	h_w
0	02110a04	02110a04	– –
1	28000010	28000010	4 4
2	00400000	00400000	2 2
3	80008000	80008000	0 0
$\mathcal{L}$	00000000	80008000	0 0
4	00000000	81008102	0 1
5	00000000	8000840a	0 2
6	00000000	850a9520	0 4
$\mathcal{L}$	850a9520	00000000	0 0
$\varSigma$			19

Rd.	ΔL_i	ΔR_i	h_w
0	02110a04	00000000	– –
1	28000010	00000000	4 0
2	00400000	00000000	2 0
3	80008000	80008000	0 0
$\mathcal{L}$	00000000	80008000	0 0
3	81008102	81008102	1 1
4	8000840a	8000840a	2 2
5	850a9520	2a102a10	4 4
$\mathcal{L}$	2a102a10	850a9520	0 0
$\varSigma$			20

Rd.	ΔL_i	ΔR_i	h_w
0	28000010		–
1	00400000	00400000	2 –
2	80008000	80008000	0 0
$\mathcal{L}$	00000000	80008000	0 0
3	00000000	81008102	0 1
4	00000000	8000840a	0 2
5	00000000	850a9520	0 4
$\mathcal{L}$	850a9520	00000000	0 0
$\varSigma$			9

Rd.	ΔL_i	ΔR_i	h_w
0	28000010		–
1	00400000	00000000	2 –
2	80008000	80008000	0 0
$\mathcal{L}$	00000000	80008000	0 0
3	81008102	81008102	1 1
4	8000840a	8000840a	2 2
5	850a9520	850a9520	4 4
$\mathcal{L}$	2a102a10	850a9520	0 0
$\varSigma$			16

Rd.	ΔL_i	ΔR_i	h_w
0	00000000	02110a04	– –
1	00000000	28000010	0 4
2	00000000	00400000	0 2
3	00000000	80008000	0 0
$\mathcal{L}$	80008000	00000000	0 0
4	81008102	00000000	1 0
5	8000840a	00000000	2 0
6	850a9520	00000000	4 0
$\mathcal{L}$	af1abf30	850a9520	0 0
$\varSigma$			13

Rd.	ΔL_i	ΔR_i	h_w
0	28000010	28000010	– –
1	00400000	00400000	2 2
2	80008000	80008000	0 0
3	83008302	81008102	2 1
$\mathcal{L}$	00000000	83008302	0 0
4	00000000	80088c02	0 5
5	00000000	8502b508	0 5
6	00000000	d0020420	0 7
$\mathcal{L}$	d0020420	00000000	0 0
7	00801000	00000000	4 0
8	10015001	00000000	2 0
9	52211224	00000000	5 0
$\mathcal{L}$	57611764	52211224	0 0
$\varSigma$			35

Rd.	ΔL_i	ΔR_i	h_w
0	00000000	00508402	– –
1	00000000	24023408	0 4
2	00000000	50c080e0	0 7
3	00000000	01810203	0 5
$\mathcal{L}$	01810203	00000000	0 0
4	000c0800	00000000	5 0
5	20000000	00000000	3 0
6	00400040	00000000	1 0
$\mathcal{L}$	00400040	00400040	0 0
7	80408140	80408140	2 2
8	00400542	00400542	3 3
9	8542904a	8542904a	4 4
$\mathcal{L}$	08150815	8542904a	0 0
$\varSigma$			37

Rd.	ΔL_i	ΔR_i	h_w
0	00000000		–
1	00000000	0a204205	0 –
2	00000000	02110a04	0 5
$\mathcal{L}$	02110a04	00000000	0 0
3	28000010	00000000	4 0
4	00400000	00000000	2 0
5	80008000	80008000	0 0
$\mathcal{L}$	00000000	80008000	0 0
6	81008102	81008102	1 1
7	8000840a	8000840a	2 2
8	850a9520	850a9520	4 4
$\mathcal{L}$	2a102a10	850a9520	0 0
$\varSigma$			25

optimal differential characteristics[3]. In this case, the problem to find optimal differential characteristics is modeled as a Boolean satisfiability problem, and can then be solved by a SAT solver. As the differential model of a cipher can be rather complex, we model the problem as a more general SMT (Satisfiability Modular Theories) problem. The difference to SAT problems is that SMT problems can express richer languages where e.g., sets of variables can be expressed as predicates or the problem can be modeled on word level. We describe the differential behavior of Sparx using the CVC language. This allows us to define specific constraints that can be used to limit the search space for the SAT solver.

[3] The differential models for Sparx are available at: `https://github.com/TheBananaMan/sparx-differential-attacks`

The solver then tries to find all possible valid differential characteristics for the given parameters with increasing probability.

Table 3 shows an optimal six-round differential trail. Note that h_w denotes $h_w = -\log_2(p)$, for the differential probability p through a round. One can observe that optimal differential characteristics for Sparx-64 possess an hourglass structure, i.e., the number of active bits is minimal in the middle and increases outwards. Using the probability of the best characteristic is often assumed to be an adequate approximation of the probability of the best differential. However, this approximation is not always sufficiently accurate for ARX-based ciphers. Therefore, we tried to evaluate the probability of differentials where feasible. For the best differentials for Sparx-64, we provide an overview in Table 4.

Types of Differential Characteristics. After searching differentials incrementality for a given interval of rounds, we searched for optimal characteristics among the following types:

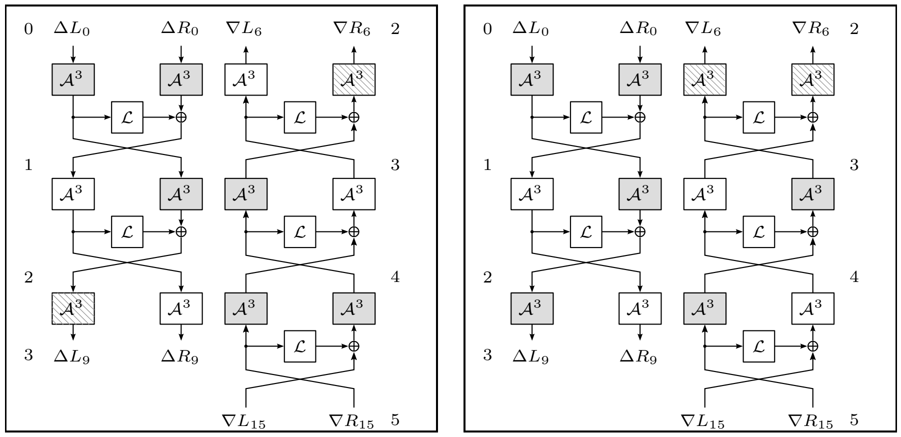
- **Type 1a.** Arbitrary single-step characteristics.
- **Type 1b.** Single-step characteristics with two active branches that have a single active branch after the step.
- **Type 1c.** Single-step characteristics with two active branches that have a single active branch before the step.
- **Type 1d.** Single-step characteristics with two active branches that have a single active branch before and afterwards.

The first category consists of single-step characteristics. The best characteristic for single-steps is a *Type 1a* characteristic with the left branch all zeros. *Type 1d* is especially interesting for our truncated differential attack.

- **Type 2a.** Two-step top characteristics which collide after the XOR in the right branch after the first step.
- **Type 2b.** Two-step bottom characteristics with only the left branch active at the first step.
- **Type 2c.** Two-step characteristics where only the left branch is active in the first, and threfore only the right branch is active in the second step.
- **Type 2d.** 4.5-round versions of Type 2a, but only two rounds before the collision for the left and one round before for the right branch.
- **Type 2e.** For the single-sided-top type, we further investigated the versions of Type 2a where the first step covers only one round.

The second category consists of two-step characteristics, that are also used in our boomerang/rectangle distinguishers. We use the two-step characteristic of *Type 2c* for the top trail and *Type 2b* for the bottom trail of our rectangle distinguisher. We further considered three-step characteristics for boomerang/rectangle attacks in our third category:

- **Type 3a.** Three-step characteristics where both branches are active in the first step, and only one branch is active in the subsequent steps, as is used in both top and bottom trail of the single-sided bottom type of boomerang.



(a) Free-middle boomerang **(b)** Single-sided-bottom boomerang

Fig. 2: Types of five-step boomerangs. White $\mathcal{A}^3$ boxes are inactive (zero difference); gray $\mathcal{A}$-boxes are active (non- zero difference). Hatched boxes indicate active branches that do not have to be taken into account at the switch.

- **Type 3b.** Three-step characteristics where the first two steps are of Type 3a, and both branches are active in the third step.
- **Type 3c.** 7.5-round versions of Type 3b, where only one round is considered for the first step.

Our results for the best characteristics found are summarized in Table 5.

4.2 Boomerangs

From the combination of the best identified characteristics, we continued to form boomerangs. We considered three types of boomerangs over five steps.

- **Free middle.** This type exploits that we can obtain the middle step for free if we choose our top and bottom trails such that one of them possesses a zero difference in the left branch, and the other one has a zero difference in the right branch, which is a direct application of the Ladder switch. We can obtain a five-step boomerang in this way, but have active differences in both branches in the first and in the fifth step of the wrapping rounds.
- **Single-sided bottom.** This type has both branches active at the start of the top trail, but only one active branch at the end of the bottom trail.
- **Single-sided top.** This type has both branches active at the end of the bottom trail, but only one active branch at the beginning of the top trail.

As examples, the former two types are visualized in Figure 2.

It became clear that free-middle boomerangs allowed higher probabilities. Table 6 summarizes the best boomerang that consist of a single characteristic that we could find for one up to five steps. Through a single step, there exist various boomerangs with probability one:

$$\Pr\left[(\Delta L_0, \Delta R_0) \xrightarrow{\text{1 step}} (\Delta L_3, \Delta R_3)\right] = 1,$$

for all characteristics with $\Delta L_0 = 0$ and $\Delta L_3 = \mathcal{L}(\Delta R_3)$; alternatively, it also holds for all characteristics with $\Delta R_0 = \Delta R_3 = 0$.

Over two steps, there exist two-step boomerangs with

$$\Pr\left[(\Delta L_0, \Delta R_0) \xrightarrow{\text{2 steps}} (\Delta L_6, \Delta R_6)\right] \geq 2^{-6},$$

namely for characteristics of the form

- $\Delta L_0 = 0$ and $\Delta R_0 \in \{\texttt{28000010}, \texttt{00400000}\}$ and $\Delta L_6 = \mathcal{L}(\Delta R_6)$, or
- $\Delta R_0 = 0$ and $\Delta R_6 \in \{\texttt{81008102}, \texttt{8000840a}\}$ and $\Delta L_6 = \mathcal{L}(\Delta R_6)$.

For three steps, the best boomerangs have probability 2^{-12}, using the single-step characteristic with the highest probability of Type 1a for the top trail, and a similar characteristic mirrored vertically and starting from the bottom difference $(\Delta L_9, \Delta R_9) = (\texttt{83008302}, \texttt{81008102})$. Similarly, we obtain from the combination of the characteristics of Type 2a and Type 1a boomerangs with probability of 2^{-44} over four steps. Over five steps, the highest probability of a boomerang with fixed characteristics results from combining a characteristic of Type 2a with the highest probability at the top with a characteristic of Type 2b with the highest probability at the bottom.

Near-optimal Differential Trails. Boomerangs that employ a single characteristic are of limited expressiveness as we noticed strong differential clustering effects in Sparx. For boomerangs, they are particularly strong in the switching rounds. Our purpose was to find good boomerangs of five steps, where we focused on the free-middle approach. We used the best characteristics of Type 1b and Type 2a as top and Type 1a and Type 2b as bottom trails as a base to study their probability empirically over a feasible subset of the three steps in the middle. Moreover, our automated search for optimal differential characteristics yielded many near-optimal differentials with probability slightly smaller than that of the optimal ones; as one could anticipate, this small change in the probability stemmed from the fact that bits adjacent to the active bits in the optimal differentials were also active in the near-optimal ones, mainly in the first or the last round. Hence, we also considered those near-optimal trails in our investigation of potential start and end differences for boomerangs. The subset of our results is given in Table 7. We used a variant of them for our rectangle attack in Section 6.

5 Truncated-Differential Attack on Sparx-64/128

High-Level View. This section describes a truncated-differential attack on 16-round Sparx-64/128. On a high level, the Feistel-like structure allows generic

Table 6: Best found boomerangs on step-reduced SPARX-64/128; for up to three steps, we verified them experimentally with 100 random keys and 2^{20} random pairs each. Values in parentheses are products of the empirical probabilities over the three steps in the middle from Table 7 with the theoretical probabilities over the remaining step(s).

#Steps	Input difference		Output difference		h_w	
s	ΔL_0	ΔR_0	ΔL_{3s}	ΔR_{3s}	theor.	empiric.
1	00000000	00400000	83008302	81008102	0	0
2	00000000	28000010	8000840a	00000000	6	5.11
2	00000000	28000010	81008102	00000000	6	5.16
2	00000000	28000010	850a9520	00000000	6	5.31
3	00000000	28000010	83008302	81008102	12	10.55
3	00000000	28000010	8a048e0e	8000840a	12	11.43
4	02110a04	02110a04	83008302	81008102	44	(40.34)
5	28000010	28000010	2a102a10	850a9520	78	(68.54)
5	02110a04	02110a04	2a102a10	850a9520	76	(72.18)

Table 7: Experimental probabilities of free-middle boomerangs over three steps. Each value represents $-\log_2(p)$, where p is the average probability of correct quartets from 100 test runs of random independent keys with 2^{30} random text pairs each.

	$(\Delta L_0, \Delta R_0)$	
$(\Delta L_9, \Delta R_9)$	(00000000, 80008000)	(00000000, 81008102)
(80008000, 80008000)	20.18	26.54
(83008302, 81008102)	16.34	22.74
(40404000, 00400000)	27.21	30.32

trails that pass through almost two steps so that only one branch is active. The core observation of our attack is the existence of Type 1d differentials, i.e., trails that have an inactive branch before and after a step with probability $\gg 2^{-32}$. One such trail is illustrated in Table 8. The trail is truncated after Round 9; thereupon, its precise differences are irrelevant as long as it will cancel in the right branch after the linear layer, and the zero-difference branch can propagate through two further steps (i.e. Rounds 13-18 in Table 8). Thus, an adversary can observe that only a single branch will be active after five steps; tThe final linear layer can then be easily inverted. On the downside, the probability of truncated trails must exceed 2^{-32} for a useful distinguisher.

To ensure a sufficient probability of the differential, we employ Property 1 at the plaintext side to reduce the number of steps to trace through. So, we obtain the round keys of Round 3, 4, and that for the right branch of Round 5 from guessing only 64 bits of key material. At the ciphertext side, we choose structures of 2^{32} texts, such that all texts in a structure have a constant value in the right branch, and iterate over all values on the left branch through Rounds 16-18. In

Table 8: The truncated differential trail through 16 rounds. A * symbol marks a truncated difference which can take any possible value.

Rd. i	ΔL_i	ΔR_i	h_w	Rd. i	ΔL_i	ΔR_i	h_w	Rd. i	ΔL_i	ΔR_i	h_w
				7	28000010	00000000	4 0	13	00000000	********	0 ?
2	********	********	– –	8	00400000	00000000	2 0	14	00000000	********	0 ?
3	********	********	– –	9	80008000	00000000	0 0	15	00000000	********	0 ?
$\mathcal{L}$	00000000	********	– –	$\mathcal{L}$	80008000	80008000	0 0	$\mathcal{L}$	********	00000000	0 0
4	00000000	********	– –	10	********	********	? ?	16	********	00000000	? 0
5	00000000	0a204205	0 –	11	********	********	? ?	17	********	00000000	? 0
6	00000000	02110a04	0 5	12	********	********	? ?	18	********	00000000	? 0
$\mathcal{L}$	02110a04	00000000	0 0	$\mathcal{L}$	00000000	********	0 0	$\mathcal{L}$	********	********	0 0

the following, we mount a chosen-ciphertext attack on 16-round Sparx-64/128 covering Rounds 3 through 18; the used differential trail is given in Table 8.

Structures and Sets. We choose 2^m structures of 2^{32} ciphertexts each from a base text $S^0_{18} = (L_{18}, R_{18})$, and $2^{32} - 1$ derived texts $S^i_{18} = (L^i_{18}, R_{18})$ from iterating over all 2^{32} values L_{18}, and derive the 2^{32} ciphertexts $C^i \leftarrow \mathcal{L}(S^i)$ that form the structure. Since we employ all 2^{32} possible values for the right branch of Rounds 16 to 18, their 2^{63} pairs will form all possible differences in this branch about 2^{31} times at any point until the end of Round 12, i.e., Δ_{12}. From experiments, we observed that the truncated differential (80008000, 80008000) leads to (00000000, ********) with probability $2^{-17.36}$. Hence, there is a subset of *good* differences Δ_{12} that can lead to (80008000, 80008000) with this accumulated probability. Since we have 2^{31} pairs for each such Δ_{12}, we expect that there are about $2^{31-17.36} \approx 2^{13.64}$ pairs with $\Delta_9 =$ (80008000, 80008000), and $2^{13.64-6-5} = 2^{2.64}$ pairs that follow our trail up to Δ_5. We have approximately 2^{63} pairs in a structure that have our desired difference with probability 2^{-64}, so we expect 2^{-1} false positive pairs from the structure.

Experimental Verification. We verified a variant of our distinguisher experimentally using 100 random keys and 2^{32} random pairs. For practicality, we considered it in encryption direction, i.e., we chose random pairs with start difference $(\Delta L_5, \Delta R_5) =$ (00000000, 0a204205), encrypted them to the states after Round 18 and inverted the final linear layer. On average, we obtained $2^{3.75}$ pairs with zero difference in the right branch, which corresponds to a probability of $2^{3.75-32} = 2^{-28.25}$, which is close to the expected $2^{-28.36}$.

Attack Steps. Using Property 2, we define a linear function $F : \{0,1\}^{32} \times \{0,1\}^{32} \to \{0,1\}^{64}$ with rank $n-1 = 63$, so that $F(\Delta) = 0^{64}$ for $\Delta =$ (00000000, 0a204205). The attack consists of the following steps:

1. Construct 2^m structures as described above. For each structure, request the corresponding 2^{32} plaintexts P^i from a 16-round decryption oracle.

2. Initialize a list $\mathcal{K}$ of 2^{64} key counters.
3. For each of the 2^{64} guesses of $K_2^0, K_2^1, K_2^2, K_2^3$:
   3.1 Re-encrypt all plaintexts over one round until the state after the linear layer of Round 3 and store them in $\mathcal{H}$ according to the values of their left branches. Only consider pairs that collide in $\mathcal{H}'$, which represent pairs that will collide in L_3 after the application of the linear layer $\mathcal{L}$.
   3.2 For all texts, compute (L_3, R_5), apply $F(R_r)$, and store the updated states in $\mathcal{H}$. Discard all pairs that do not collide. For each colliding pair, increment the counter for the current key candidate in $\mathcal{K}$.
4. Output the keys in descending order of their corresponding counters.

Complexity. The computational complexity results from:

– **Step 1** requires 2^{m+32} 16-round decryption. We assume the computational costs for a decryption and encryption are equal.
– **Step 3.1** requires $2^{64} \cdot 2^{m+32} \cdot 1/16 \cdot 2 \approx 2^{m+92}$ encryption equivalents since we consider one out of 16 rounds. From the $\binom{2^{32}}{2} \approx 2^{63}$ pairs of one structure, we expect $2^{63-32} = 2^{31}$ false positive pairs for each structure at this step.
– We approximate the costs for a call to F by those of a call to two Speckey rounds since both branches are used. The complexity of **Step 3.2** is therefore given by $2^{64} \cdot 2^{31+m} \cdot 4/32 \approx 2^{m+92}$ encryption equivalents on average. We expect about $2^{63-64} = 2^{-1}$ false-positive pairs per structure and key candidate, whereas we have $2^{31-28.36} \approx 2^{2.64}$ correct pairs for the correct key candidate, again per structure.

The computational complexity sums to

$$2^{m+32} + 2^{m+92} + 2^{m+91} \approx 2^{m+93} \text{ Encryptions.}$$

The memory complexity stems from storing a byte counter for the current key candidate, i.e., $2^{64} \cdot 8/64 = 2^{61}$ states, plus 2^{32} texts. The data complexity is given by 2^{m+32}. A single structure, i.e., $m = 1$, suffices to obtain at least two correct pairs for the correct key.

6 Rectangle Attack on 16-round Sparx-64/128

High-Level View. This section describes a rectangle attack on 16-round Sparx-64/128. Our attack starts after the second round of the cipher, i.e., it starts with Round 3. Again, we guess 64 key bits to get through Rounds 3 and 4 and the right branch of Round 5. The attack covers then Rounds 3 through 18.

Differential Trails. Table 9 illustrates the employed differential trails. The top trail covers Rounds 3 through 9 and the right part of Rounds 10 to 12 since the right part contains a zero difference which propagates for free through the $\mathcal{A}^3$ box of Rounds 10 to 12. The bottom trail covers Rounds 13 through 18, and the left part of Rounds 10 through 12 in decryption direction. Again, the bottom

Table 9: Our used differential characteristic through the top (left) and bottom (right) trail for our 16-round rectangle attack on Sparx-64/128.

Rd. i	ΔL_i	ΔR_i	h_w	Rd. i	ΔL_i	ΔR_i	h_w
4	28000010		– –	10	00000000	********	0 –
5	00400000	00400000	2 –	11	00000000	********	0 –
6	80008000	80008000	0 0	12	00000000	********	0 –
$\mathcal{L}$	00000000	80008000	0 0	$\mathcal{L}$	02110a04	00000000	0 0
7	00000000	********	0 –	13	********	00000000	– 0
8	00000000	********	0 –	14	********	00000000	– 0
9	00000000	********	0 –	15	********	00000000	– 0
$\mathcal{L}$	********	00000000	0 –	$\mathcal{L}$	80008000	80008000	0 0
10	********	00000000	– 0	16	81008102	81008102	1 1
11	********	00000000	– 0	17	8000840a	8000840a	2 2
12	********	00000000	– 0	18	850a9520	850a9520	4 4
$\mathcal{L}$	********	********	– 0	$\mathcal{L}$	2a102a10	850a9520	0 0

trail has a zero difference in that part in the bottom trail which propagates for free through the $\mathcal{A}^3$ box until the begin of Round 10.

Again, we experimentally verified the boomerang switch in the middle. From 100 experiments with random keys and 2^{26} independently at random chosen pairs (P, P') with difference $\alpha = (\texttt{80008000}, \texttt{80008000})$, encrypted through three steps to (C, C'), applied the δ-shift $(\texttt{80008000}, \texttt{80008000})$ to obtain (D, D'), decrypted those back to (Q, Q'), and counted the number of times that $Q \oplus Q' = \alpha$. We observed an average probability of approximately $2^{-20.18}$. So, for the correct key, we obtain a probability of approximately $(\widehat{pq})^2 \approx \left(2^{-2}\right)^2 \cdot 2^{-20.18} \cdot \left(2^{-14}\right)^2 \approx 2^{-52.18}$ for a valid quartet.

Attack Procedure. Choose a linear function $F : \{0,1\}^{64} \to \{0,1\}^{64}$ of rank 63 s.t. $F(\Delta L_4 \,\|\, \Delta R_5) = 0^{64}$. The attack consists of the following steps:

1. Initialize a list of key counters $\mathcal{L}$ to zero, for all 2^{64} possible values for the round keys of Round 2. Initialize two empty hash maps $\mathcal{Q}$ and $\mathcal{R}$.
2. Choose 2^m ciphertext pairs (C, D) with difference $(\texttt{2a102a10}, \texttt{850a9520})$, and ask for their corresponding plaintexts (P, Q). Store the pairs into $\mathcal{H}$ indexed by P.
3. For each of the 2^{64} guesses of $(K_2^0, K_2^1, K_2^2, K_2^3)$:
   - 3.1 Partially re-encrypt all plaintext pairs (P, Q) to their corresponding states (L_4^P, R_5^P) and (L_4^Q, R_5^Q).
   - 3.2 Apply $F((L_4, R_5))$ to all states and store the corresponding outputs $(\widehat{L}_4^P, \widehat{R}_5^P)$ and $(\widehat{L}_4^Q, \widehat{R}_5^Q)$ into a hash table $\mathcal{Q}$. Only consider pairs of pairs $p = (\widehat{L}_4^P, \widehat{R}_5^P)$, $q = (\widehat{L}_4^Q, \widehat{R}_5^Q)$, $p' = (\widehat{L}_4^{P'}, \widehat{R}_5^{P'})$, $q' = (\widehat{L}_4^{Q'}, \widehat{R}_5^{Q'})$ that collide in either $(p, q) = (p', q')$ or $(p, q) = (q', p')$ and discard all further quartets. We expect $2^{2m} \cdot 2^{2 \cdot -64} \approx 2^{2m-128}$ quartets on average.

3.3 If a quartet survives, increment the counter for the current key guess. Choose a plaintext pair with our desired difference – w.l.o.g., (p, p') – from the current quartet, and check for all remaining key bits if it follows our path until Round 6. If yes, encrypt it further roundwise until Round 9. If all roundwise checks pass, check for p if it encrypts to ciphertext C. If yes, test again for (q, q') and output the key candidate if it also matches.

4. If no key candidate has been returned, return $\perp$.

For $m = 58.6$ pairs, we can expect $(2^m \widehat{pq})^2/2^n \approx 2^{117.2} \cdot 2^{-52.18}/2^{64} \approx 2$ valid quartets for the correct key guess. In contrast, we can expect $2^{117.2-2\cdot 64} = 2^{-10.8}$ quartets for a wrong key guess.

Complexity. The computational complexity results from:

- **Step 2** requires $2 \cdot 2^{58.6} \approx 2^{59.6}$ 16-round decryptions. We assume that the computational costs for a decryption and encryption are equal.
- **Steps 3.1 and 3.2** require $2^{64} \cdot 2 \cdot 2^m \cdot 6/32 \approx 2^{122.2}$ encryption equivalents since we consider five out of 32 Speckey rounds in the 16-round cipher for re-encryption and approximate the costs for computing F by the costs of a Speckey round.
- **Step 3.2** will require $2^{64} \cdot 2 \cdot 2^m = 2^{m+65}$ memory accesses (MAs) and comparisons.
- **Step 3.3** will require at most $2^{64} \cdot 2^{2m-128} \cdot 2^{64} \approx 2^{117.2}$ encryption equivalents to identify the correct key.

Hence, the computations are upper bounded by approximately

$$2^{59.6} + 2^{122.2} \approx 2^{122.2} \text{ encryptions} \quad \text{and} \quad 2^{59.6} + 2^{123.6} \approx 2^{123.6} \text{ MAs.}$$

The data complexity is upper bounded by $2^{59.6}$ chosen ciphertexts. The memory complexity is upper bounded by storing at most $4 \cdot 2^{59.6}$ states at a time, which is equivalent to storing approximately $2^{61.6}$ states.

7 Conclusion

This work presents two standard differential attacks using truncated differentials and rectangle attacks on 16-round Sparx-64/128. The former attack builds upon a nine-round (three-step) differential trail that is extended by a six-round (two-step) truncated trail. Adopting the observation by Abdelkhalek et al. [1], we can turn the distinguishers in a 16-round chosen-ciphertext attack and recover the round keys by just guessing 64-bit of the key material. Our truncated differential attack requires approximately 2^{32} chosen ciphertexts, about 2^{32} states, and approximately 2^{93} encryption equivalents. Our proposed rectangle attack exploits the Feistel structure of Sparx using differential trails with inactive branches over their middle step; similarly, the yoyo attack in the Appendix profits from the structure over the end. It may be an interesting subject for further studies to investigate yoyo cycles of more iterations of en- and decryption.

We stress that our attacks do not threaten the security of Sparx-64/128, but provide deeper insights in its security against attacks in the single-key setting. We can observe a strong clustering effects of many differential characteristics in our studies and exploit them in all our attacks; it remains subject to further studies to employ them for further rounds. For public verification and future works, our trails, tests, and implementations of Sparx-64/128 will be published into the public domain[4].

References

1. Ahmed Abdelkhalek, Mohamed Tolba, and Amr M. Youssef. Impossible Differential Attack on Reduced Round SPARX-64/128. In Marc Joye and Abderrahmane Nitaj, editors, *AFRICACRYPT*, volume 10239 of *Lecture Notes in Computer Science*, pages 135–146, 2017.
2. Eli Biham, Alex Biryukov, Orr Dunkelman, Eran Richardson, and Adi Shamir. Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR. In Stafford E. Tavares and Henk Meijer, editors, *SAC*, volume 1556 of *Lecture Notes in Computer Science*, pages 362–376. Springer, 1998.
3. Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2001.
4. Eli Biham, Orr Dunkelman, and Nathan Keller. New Results on Boomerang and Rectangle Attacks. In Joan Daemen and Vincent Rijmen, editors, *FSE*, volume 2365 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.
5. Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.
6. Alex Biryukov, Gaëtan Leurent, and Léo Perrin. Cryptanalysis of Feistel Networks with Secret Round Functions. In Orr Dunkelman and Liam Keliher, editors, *SAC*, volume 9566 of *Lecture Notes in Computer Science*, pages 102–121. Springer, 2015.
7. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang Connectivity Table (BCT) for Boomerang Attack. In *EUROCRYPT*, Lecture Notes in Computer Science, 2018. to appear.
8. Daniel Dinu, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Johann Großschädl, and Alex Biryukov. Design Strategies for ARX with Provable Bounds: Sparx and LAX. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT I*, volume 10031 of *Lecture Notes in Computer Science*, pages 484–513, 2016.
9. Joan Daemen and Michaël Peeters and Gilles Van Assche and Vincent Rijmen. Nessie Proposal: NOEKEON, 2000. `http://gro.noekeon.org/Noekeon-spec.pdf`.
10. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In Bruce Schneier, editor, *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2000.
11. Gaëtan Leurent. Improved differential-linear cryptanalysis of 7-round chaskey with partitioning. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT I*, volume 9665 of *Lecture Notes in Computer Science*, pages 344–371. Springer, 2016.

[4] `https://github.com/TheBananaMan/sparx-differential-attacks`

12. Mate Soos. CryptoMiniSat SAT solver. `https://github.com/msoos/cryptominisat/`.
13. Sondre Rønjom, Navid Ghaedi Bardeh, and Tor Helleseth. Yoyo Tricks with AES. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT I*, volume 10624 of *Lecture Notes in Computer Science*, pages 217–243. Springer, 2017.
14. Stefan Kölbl. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. `https://github.com/kste/cryptosmt`.
15. Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef. Multidimensional Zero-Correlation Linear Cryptanalysis of Reduced Round SPARX-128. In Carlisle Adams and Jan Camenisch, editors, *SAC*, volume 10719 of *Lecture Notes in Computer Science*, pages 423–441. Springer, 2017.
16. Vijay Ganesh and Trevor Hansen and Mate Soos and Dan Liew and Ryan Govostes. STP constraint solver. `https://github.com/stp/stp`.
17. David Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 1999.

Supplementary Material

A Yoyo Attack

The Concept of Yoyo Cryptanalysis. Yoyo attacks are closely related to boomerangs. In both techniques, the adversary first lets the oracle encrypt chosen texts, observes the corresponding encryptions and adaptively chooses new ciphertexts that are then decrypted in the hope for a certain property in their corresponding plaintexts. Initially, yoyo attacks have been proposed by Biham et al. on Skipjack-3XOR [2], and have been revived by Biryukov et al. [6] for analyzing Feistel networks with secret round functions. Recently, Rønjom et al. [13] showed yoyo-based distinguishers and key-recovery attacks on generic SPNs and applications to round-reduced AES.

Assume, $E : \mathbb{F}_2^{m \cdot b} \to \mathbb{F}_2^{m \cdot b}$ is a permutation over elements of $n = m \cdot b$ bits that can be divided into m words of b bits each. The core idea of yoyos is to encrypt chosen plaintext pairs (P^0, P^1) with a certain input difference through E and collect the corresponding ciphertexts (C^0, C^1). The cryptanalyst crafts a set of new ciphertexts from the recombination of words from C^0 and C^1. Let $C^0[i]$ denote the i-th b-bit word of C^0. Let $v = (v_0, \ldots, v_{m-1})$ denote a Boolean vector, i.e., $v \in \mathbb{F}_2^m$, and let $\rho : \mathbb{F}_2^{m \cdot b} \times \mathbb{F}_2^{m \cdot b} \times \mathbb{F}_2^m \to \mathbb{F}_2^{m \cdot b}$ be defined as

$$\rho\left(C^0, C^1, v\right) \stackrel{\text{def}}{=} \left(C^{v_0}[0] \parallel \ldots \parallel C^{v_{m-1}}[m-1]\right).$$

Given $v \neq (0, 0, \ldots, 0)$, one can derive two modified ciphertexts $C^2 = \rho(C^0, C^1, v)$ and $C^3 = \rho(C^0, C^1, \overline{v})$, where $\overline{v}$ denotes the vector of the element-wise inverse Boolean vector, i.e., $\overline{v}_i = 1 \oplus v_i$ for $0 \leq i < m$. Then, C^2 consists of the words of C^0 at all positions i where $v_i = 0$ and of the words of C^1 at the remaining positions. The situation is vice versa for C^3. The cryptanalyst then queries the so-derived ciphertexts to obtain the corresponding plaintexts (P^2, P^3). The core observation is that the vector v defines a word-wise difference pattern: C^2 differs from C^0 in exactly the words where $v_i = 1$ holds. This difference pattern is preserved through any partial map of E that operates on the b-bit words separately, e.g., an S-box layer in an SPN.

Application to Sparx-64/128. A step of Sparx-64/128 can be modelled as the application of a key-dependent non-linear layer of $\mathcal{A}^3$ boxes that transform the 32-bit branches individually, followed by a linear layer $\mathcal{L}$ that mixes one branch into the other one. Hence, it is senseful to model $v \in \mathbb{F}_2^2$ as a two- element Boolean vector. Let $C^0 = (L_r^0, R_r^0)$ and $C^1 = (L_r^1, R_r^1)$ denote two ciphertexts after r rounds of Sparx-64/128. Let $S_r^0 = L_r^0 \oplus \mathcal{L}(R_r^0)$ and $S_r^1 = L_r^1 \oplus \mathcal{L}(R_r^1)$ denote the words before the linear layer of the r-th step. We focus on ciphertexts with pairwise distinct branches before the linear layer: $S_r^0 \neq S_r^1$ and $R_r^0 \neq R_r^1$. Our goal is to derive two new ciphertexts

$$C^2 = (S_r^2, R_r^2) = \rho_C(C^0, C^1, (0, 1)) = (S_r^0, R_r^1)$$
$$C^3 = (S_r^3, R_r^3) = \rho_C(C^0, C^1, (1, 0)) = (S_r^1, R_r^0).$$

For this purpose, we define the ciphertext-recombination function ρ_C in order to invert the last linear layer as

$$\rho_C(C^0, C^1, v) \overset{\text{def}}{=} \left(L_r^{v_0} \oplus \mathcal{L}(R_r^0) \oplus \mathcal{L}(R_r^1), R_r^{v_1}\right).$$

Since the $\mathcal{A}^3$ boxes preserve the difference pattern of v, we obtain then $(L_{r-1}^2, R_{r-1}^2) = (L_{r-1}^0, R_{r-1}^1)$ and $(L_{r-1}^2, R_{r-1}^2) = (L_{r-1}^1, R_{r-1}^0)$. Clearly, the differences are preserved through the inverse final round:

$$\left(L_{r-1}^0, R_{r-1}^0\right) \oplus \left(L_{r-1}^1, R_{r-1}^1\right) = \left(L_{r-1}^2, R_{r-1}^2\right) \oplus \left(L_{r-1}^3, R_{r-1}^3\right).$$

Moreover, assume that both ciphertexts shared the same state $S_{r-1}^0 = S_{r-1}^1$, which meant that the right branch had a zero difference through the $(r-1)$-th step: $S_{r-1}^0 = \mathcal{L}(R_{r-1}^0) \oplus L_{r-1}^0 = \mathcal{L}(R_{r-1}^1) \oplus L_{r-1}^1 = S_{r-1}^1$. We can reformulate this to $\mathcal{L}(R_{r-1}^0) \oplus L_{r-1}^0 \oplus \mathcal{L}(R_{r-1}^1) \oplus L_{r-1}^1 = 0$. Then, this collision is also preserved between the derived ciphertexts

$$S_{r-1}^2 = \mathcal{L}(R_{r-1}^2) \oplus L_{r-1}^2 = \mathcal{L}(R_{r-1}^1) \oplus L_{r-1}^0 = \mathcal{L}(R_{r-1}^0) \oplus L_{r-1}^1 = S_{r-1}^3.$$

It follows that the left branch had a zero difference through the $(r-2)$-th step. However, since we assumed that the branches are pair-wise distinct, the values $\neq S_{r-1}^0 = S_{r-1}^1$ differ, which means that $L_{r-2}^2 = L_{r-2}^3 \neq L_{r-2}^0 = L_{r-2}^1$ differ. Therefore, although the differences will be preserved after two inverse steps, $S_{r-2}^2 \oplus S_{r-2}^3 = S_{r-2}^0 \oplus S_{r-2}^1$, the values S will differ. Therefore, the difference in the right branch through the $(r-2)$-th step is not guaranteed to be preserved. Hence, (C^2, C^3) follows the same differential path as (C^0, C^1) through 2.5-steps in decryption direction with probability one if $S_{r-1}^0 \oplus S_{r-1}^1$ holds. A similar property also holds in encryption direction. Define a simpler plaintext-recombination function $\rho_P(P^0, P^1, v) \overset{\text{def}}{=} (L_0^{v_0}, R_0^{v_1})$. Given two distinct plaintexts (P^0, P^1) for which $L_1^0 = L_1^1$ holds after the first step, we can derive plaintexts $P^2 = \rho_P(P^0, P^1, (0, 1))$ and $P^3 = \rho_P(P^0, P^1, (1, 0))$ that will also follow the same differential as (P^0, P^1) through 2.5 steps: $S_2^2 \oplus S_2^3 = S_2^0 \oplus S_2^0$ and $S_3^2 \oplus S_3^3 = S_3^0 \oplus S_3^0$.

Table 10: The appended rounds 4-9 (left), top trail (middle), and bottom trail (right, Rounds 13 to 18) of our yoyo attack.

Rd. i	ΔL_i	ΔR_i	h_w	Rd. i	ΔL_i	ΔR_i	p	Rd. i	ΔL_i	ΔR_i	p
				7	00400000	00400000	2 2	13	********	00000000	– 0
3	********	********	– –	8	80008000	80008000	0 0	14	********	00000000	– 0
2	********	********	– –	9	81008102	83008302	1 2	15	********	00000000	– 0
$\mathcal{L}$	********	********	– –	$\mathcal{L}$	00000000	81008102	0 0	$\mathcal{L}$	********	********	0 0
4	0a604205	********	– –	10	00000000	8000840a	0 2	16	********	********	– 0
5	02110a04	14080000	5 –	11	00000000	850a9520	0 4	17	********	********	– 0
6	28000010	10281028	4 3	12	00000000	802ad4a8	0 6	18	********	********	– 0
$\mathcal{L}$	28000010	28000010	0 0	$\mathcal{L}$	802ad4a8	00000000	0 0	$\mathcal{L}$	********	********	0 0

Yoyo Attack. Our yoyo attack on 16 rounds starts at (including) Round 3 until the end of Round 18. Table 10 shows our distinguisher. By guessing 64 bit of the secret key, we guess our way through Rounds 3 and 4, as well as through the right branch of Round 5, as in our previous attacks. The probability that a pair of plaintexts collides after the linear layer at the end of Round 9 is 2^{-19} and 2^{-31} that it follows the differential path to $(\texttt{802ad4a8}, \texttt{00000000})$ until the end of Round 12. The differential follows a truncated path until the end of Round 18. Given a ciphertext pair (C^0, C^1), the adversary then computes $C^2 = \rho_C(C^0, C^1, (0, 1))$ and $C^3 = \rho_C(C^0, C^1, (1, 0))$ as explained above, and asks for the corresponding decryption. The probability that $(L^2_{12}, R^2_{12}) \oplus (L^3_{12}, R^3_{12}) = (\texttt{802ad4a8}, \texttt{00000000})$ holds is one; with another probability 2^{-31}, both pairs follow the inverse differential up to (P^2, P^3) and have $(L^2_4, R^2_5) \oplus (L^3_4, R^3_5) = (\texttt{0a604205}, \texttt{14080000})$. So, the probability for a real yoyo is approximately 2^{-62} whereas it is 2^{-64} for a random permutation. If a yoyo quartet is found, the current key candidate is tested with another text, and is discarded otherwise.

Complexity. The attack requires the entire codebook as chosen plaintexts. The computational complexity can be approximated by 2^{64} initial encryptions, $2^{64} \cdot 2^{63}$ lookups, where each key candidate uses 2^{62} plaintext pairs, the same number of branch swaps (computations of ρ_c), and $2^{64} \cdot 2^{62} \cdot 2^{-64} = 2^{62}$ additional lookups to test if the current key candidate is correct. The memory requirements are 2^{64} states for the codebook.