Cryptanalysis of Reduced sLiSCP Permutation in Sponge-Hash and Duplex-AE Modes

Yunwen Liu[1,2], Yu Sasaki[3(✉)], Ling Song[4,5], and Gaoli Wang[6]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
yunwen.liu@esat.kuleuven.be
[2] College of Liberal Arts and Sciences, National University of Defense Technology, Changsha, China
[3] NTT Secure Platform Laboratories, 3-9-11, Midori-cho Musashino-shi, Tokyo 180-8585, Japan
sasaki.yu@lab.ntt.co.jp
[4] Nanyang Technological University, Singapore, Singapore
[5] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
songling@iie.ac.cn
[6] Department of Cryptography and Network Security, East China Normal University, Shanghai 200062, China
glwang@sei.ecnu.edu.cn

Abstract. This paper studies security of a family of lightweight permutations sLiSCP that was proposed by AlTawy et al. at SAC 2017. sLiSCP also specifies an authenticated encryption (AE) mode and a hashing mode based on the sponge framework, however the designers' analysis focuses on the indistinguishability of the permutation, and there is no analysis for those modes. This paper presents the first analysis of reduced-step sLiSCP in the AE and hashing modes fully respecting the recommended parameters and usage by the designers. Forgery and collision attacks are presented against 6 (out of 18) steps of the AE and hashing modes. Moreover, rebound distinguishers are presented against 15 steps of the permutation. We believe that those results especially about the AE and hashing modes provide a better understanding of sLiSCP, and bring more confidence about the lightweight version sLiSCP-light.

Keywords: sLiSCP · Simeck · Permutation · Sponge · Collision Forgery

1 Introduction

Ubiquitous computing and the Internet of Things (IoT) are developing rapidly as the new computing paradigm in information technology. The deployment of small computing devices such as Radio-Frequency Identification (RFID) tags, sensor nodes and smart cards increases fast and plays an important role in various applications. At the same time, it also brings a wide range of new security and privacy concerns. These small devices demand harsh cost constraints like low memory availability, low area requirements and power consumptions, which makes it difficult to employ conventional cryptographic algorithms. Lightweight cryptography is a field of cryptography that caters for security concerns of resource-constrained devices. Dozens of symmetric-key primitives have been proposed to address the issues, such as lightweight block ciphers (LED [18], PRESENT [12], SIMON & SPECK [6], Simeck [30] etc.), lightweight hash functions (Spongent [11], Photon [17], Quark [3] etc.), lightweight stream ciphers (Grain [19], Mickey [5], Trivium [14] etc.) and lightweight authenticated encryptions (Ascon [15], Ketje-Jr [10] and NORX [4]). Meanwhile, lightweight cryptographic algorithms including PRESENT, Photon, Grain and Trivium are adopted by ISO as new standards. Recently, the National Institute of Standards and Technology of the U.S. (NIST) has started a process for standardizing lightweight authenticated encryptions with associated data (AEAD) and hashing [26].

Among the existing lightweight cryptographic algorithms, permutation-based designs are of special interest. They have an outstanding advantage for devices that have limited resources to provide multiple cryptographic functions with low overhead. In fact, encryption, authentication, hashing, and possibly pseudorandom-bit generation which are the basic functionalities required by a security protocol can be achieved by applying a cryptographic permutation in certain modes, such as Sponge [9]. Ascon, NORX and Ketje-Jr are examples of permutation-based designs to provide both encryption and authentication.

`sLiSCP` is a family of cryptographic permutations designed by AlTawy et al. and proposed at SAC 2017 [1]. It has two instances, namely, `sLiSCP`-192 and `sLiSCP`-256 which adopt a 4-branch type-2 generalized Feistel network (GFN) where the functions in GFN are instantiated with reduced-round Simeck-48/64 [30] whose secret key is replaced with a public constant. Both `sLiSCP`-192 and `sLiSCP`-256 have 18 steps. Besides, the designers use `sLiSCP` in the sponge framework to construct authenticated encryption (AE) [8] and hash functions [7]. Considering that the coming standardization activity for lightweight cryptography by NIST takes into account the designs that support both AE and hash function, security analysis of `sLiSCP` is of great interest as an example case.

Cryptanalysis is crucial for any design. The existing security analysis of `sLiSCP` by the designers focus on the indistinguishability of the permutation, and there is no analysis in the hashing and AE modes. The designers showed that impossible differential (or zero-correlation) distinguishers reach 9 steps of the `sLiSCP` permutation and zero-sum distinguishers utilizing division property [28] can achieve 17 steps of `sLiSCP`-192/256 with complexity 2^{190} (resp. 2^{255}) for `sLiSCP`-192 (resp. `sLiSCP`-256). Without rigorous cryptanalysis, it is hard to determine the most suitable number of steps. Recently, a lightweight variant of `sLiSCP`, `sLiSCP`-light [2], was proposed by the same designers, which replaces the 4-branch generalized Feistel network with the 4-branch generalized Misty structure, and the number of steps in the permutation is reduced to only 12.

Table 1. Summary of attacks against sLiSCP

Target	Version	Attacks	Steps	Time	Data	Memory	Ref.
AE	sLiSCP-192	Forgery	6/18	$2^{104.0}$	$2^{104.0}$	negl.	Sect. 4
	sLiSCP-256	Forgery	6/18	$2^{112.2}$	$2^{112.2}$	negl.	Sect. 4
	sLiSCP-192	State Recovery	6/18	$2^{105.6}$	$2^{105.6}$	negl.	Sect. 4
Hash	sLiSCP-192	Collision	6/18	$2^{69.8}$	N/A	$2^{32.1}$	Sect. 5
	sLiSCP-256	Collision	6/18	$2^{74.8}$	N/A	$2^{46.3}$	Sect. 5
Permutation	both	Imp Diff	9/18	N/A	N/A	N/A	[1]
	both	Zero Cor	9/18	N/A	N/A	N/A	[1]
	sLiSCP-192	Zero-sum	17/18	2^{190}	N/A	negl.	[1]
	sLiSCP-256	Zero-sum	17/18	2^{255}	N/A	negl.	[1]
	sLiSCP-192	Rebound	15/18	$2^{122.7}$	N/A	$2^{37.7}$	Sect. 6
	sLiSCP-256	Rebound	15/18	$2^{168.3}$	N/A	$2^{47.7}$	Sect. 6

Our Contributions. In this paper, we provide security analysis of sLiSCP, in particular, the first results of sLiSCP in AE and hashing modes. The number of attacked steps, 6, is small compared to the full steps, 18. However, 18 steps of sLiSCP uses 216 and 288 rounds of Simeck-48 and Simeck-64 to permute 192-bit and 256-bit states, respectively, which looks conservative. Indeed, the number of steps was later reduced in sLiSCP-light. We believe that our analysis helps to understand the suitable choice of the number of steps.

Our first analysis is the 6 (out of 18) steps forgery attacks in the AE mode. The attacks fully respect the limitation by the designers, i.e. we use the size and position of the inner and outer parts (or capacity and rate) according to the designer's recommendation and the nonce is never repeated. There are two versions of the AE mode; sLiSCP-192/112 and sLiSCP-256/128 that use 112-bit and 128-bit key and claim 112-bit and 128-bit security, respectively. The attack complexities are $2^{103.96}$ and $2^{112.2}$ queries for sLiSCP-192/112 and sLiSCP-256/128 respectively. Moreover, the state recovery is applied to sLiSCP-192/112.

We then convert the above attacks to find collisions in the hashing mode. The claimed security is 80 bits and 96 bits for sLiSCP-192 and sLiSCP-256 respectively, thus naively applying the attacks on AE to hashing modes is worse than the birthday attack. In the hash setting, attackers have access to the internal state value and can choose message values to control the differential propagation. To exploit this property, we use the multi-block strategy and find collisions with $2^{69.8}$ and $2^{74.8}$ computations for sLiSCP-192 and sLiSCP-256, respectively.

Finally, we evaluate sLiSCP as a permutation by applying rebound attacks [23,25]. Although the zero-sum distinguisher by the designers [1] can break more steps, their complexities are very close to the permutation size. Our rebound attacks reach only 15 rounds but the computational complexities, $2^{122.7}$ for sLiSCP-192 and $2^{168.3}$ for sLiSCP-256, are significantly smaller than the permutation size. The differential based approach can be applied to AE or hash settings and our rebound attacks provide better understandings to decide the suitable number of steps in the lightweight design e.g. sLiSCP-light. Our results are summarized in Table 1 along with the attacks by the designers.

The core of our attacks is the discovery of efficient differential trails for the sLiSCP permutation. Because of the large state size and the complex underlying Simeck permutation, it is infeasible to find useful trails with existing automated search tools.[1] In this paper, we start with our differential trail search strategy.

Paper Outline. Section 2 describes the sLiSCP specification. Section 3 explains how to search for differential trails for large sLiSCP permutations. Section 4 describes forgery and state-recovery attacks in the AE mode. Section 5 describes collision attacks in the hashing mode. Section 6 presents rebound attacks against sLiSCP permutations. We conclude this paper in Sect. 7.

2 Specification of sLiSCP

2.1 sLiSCP Permutation

The sLiSCP permutation F is denoted as sLiSCP-b, where $b = 4m$ and $m \in \{48, 64\}$. As depicted in Fig. 1, F updates the input $(X_0^0, X_1^0, X_2^0, X_3^0)$ of four m-bit words in s steps and gets the output $(X_0^s, X_1^s, X_2^s, X_3^s)$. The permutation

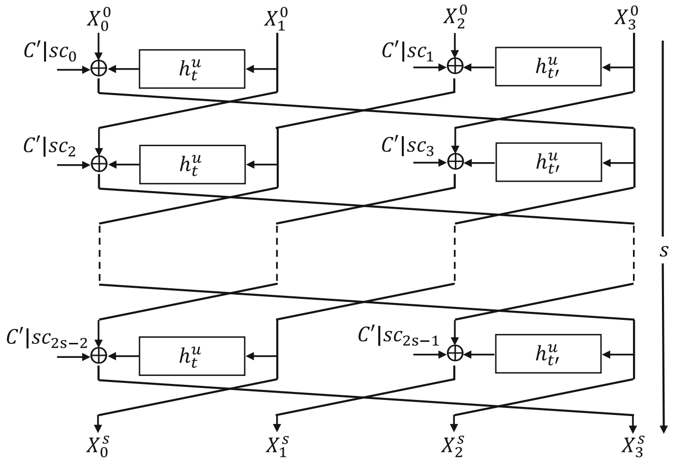


Fig. 1. sLiSCP permutation using Simeck^u-m as h_t^u

F can be described in terms of the step function f as

$$F(X_0^0, X_1^0, X_2^0, X_3^0) = f^s(X_0^0, X_1^0, X_2^0, X_3^0) = (X_0^s, X_1^s, X_2^s, X_3^s).$$

[1] We first tried to find the optimal 6-step differential trail for sLiSCP-192 with MILP. Even after 2,000,000 s (more than 23 days), we did not have any hope that the tool would finish. Searching for the optimized trail for sLiSCP-256 is even harder.

Table 2. Parameters for the permutation F in sLiSCP-192 and sLiSCP-256

Algorithm	Branch size m	Rounds u	Steps s	State size b	Total number of Simeck rounds $2us$
sLiSCP-192	48	6	18	192	216
sLiSCP-256	64	8	18	256	288

The step function f is built on a 4 branch Type-2 GFN and based on an u-round Simeck [1]. In step j $(0 \leq j \leq s-1)$, the step function $f(X_0^j, X_1^j, X_2^j, X_3^j)$ is defined as

$$\left(X_1^j, h_{t'}^u(X_3^j) \oplus X_2^j \oplus (C'|SC_{2j+1}), X_3^j, h_t^u(X_1^j) \oplus X_0^j \oplus (C'|SC_{2j})\right),$$

where C' and SC_j are a constant $2^m - 256$ and a step-dependent constant, respectively, "$|$" is a bitwise-OR. and $h_t^u(\cdot)$ is an u-round Simeck depending on the constant t. We sometimes omit t and denote the function by Simeck^u-$m(\cdot)$, which is further detailed as

$$h_t^u(x) = \text{Simeck}^u\text{-}m(x) = h_{u-1} \circ h_{u-2} \circ ... \circ h_0(x),$$

where $h_i(x) = h_i(x_0 \parallel x_1)$ is defined as follows (See also Fig. 5 in Appendix.):

$$h_i(x) = \left((x_0 \odot (x_0 \lll 5)) \oplus (x_0 \lll 1) \oplus x_1 \oplus (C|RC_i), x_0\right).$$

Here "$\oplus$," "$\odot$" and "$\lll$" denote bit-wise XOR, bitwise AND, and a left cyclic shift, respectively. x_0 and x_1 are $\frac{m}{2}$-bit words and C and RC_i are a constant defined as $2^{\frac{m}{2}} - 2$ and a round-dependent constant. The parameters for the permutation F in sLiSCP-192 and sLiSCP-256 are given in Table 2. Because the constants do not impact to our attacks, we omit the details of the constants. The schematic diagram of the s-step sLiSCP permutation instantiated with u-round Simeck-m is illustrated in Fig. 1.

2.2 sLiSCP Mode for Hash Function and Authenticated Encryption

Hash function and authenticated encryption are constructed using sLiSCP in the sponge-based modes. In order to specify the initialization, absorbing and squeezing phases conveniently, we use the following notations. For sLiSCP-192, the 192-bit state is denoted as 24-byte state as

$$(X_0, X_1, X_2, X_3) = (B_0, ..., B_5, B_6, ..., B_{11}, B_{12}, ..., B_{17}, B_{18}, ..., B_{23}),$$

where $X_i \in \mathbb{F}_2^{48}$ and $B_i \in \mathbb{F}_2^8$. For sLiSCP-256, the 256-bit state is denoted as

$$(B_0, ..., B_7, B_8, ..., B_{15}, B_{16}, ..., B_{23}, B_{24}, ..., B_{31}).$$

Initialization. In the hashing mode, the state is initialized to a constant value called IV. In the AE mode, the state is initialized to a mixture of nonce, key, and constant. Because we do not use those configurations in our attacks, we refer to [1] for the details of the initial set up.

Rate and Capacity. In the sponge-based construction, the b-bit state is divided into rate r and capacity c such that $r + c = b$. In both of the AE and hash modes, $r = 32$ and $r = 64$ are recommended when F is sLiSCP-192 and sLiSCP-256, respectively. (Accordingly, $c = 160$ and $c = 192$ for sLiSCP-192 and sLiSCP-256, respectively.) The byte positions of the rate are defined as B_i $(i = 6, 7, 18, 19)$ for sLiSCP-192 and B_i $(i = 8, 9, 10, 11, 24, 25, 26, 27)$ for sLiSCP-256.

Hash Mode. As depicted in Fig. 3 in Appendix, the message M is padded and split into blocks of r bits each. After the initialization, the message block is XORed with B_i $(i = 6, 7, 18, 19)$ and $(i = 8, 9, 10, 11, 24, 25, 26, 27)$ for sLiSCP-192-based and sLiSCP-256-based constructions, respectively, followed by the application of the permutation F. The absorbing phase finishes when all message blocks are processed. Then in the squeezing phase, extraction of the r' bits of the state and application of F is iterated until the entire digest is obtained. r' is recommended as $r' = 32$ for sLiSCP-192 and $r' \in \{32, 64\}$ for sLiSCP-256.

AE Mode. Firstly, the key K, the message M and the associated data A are padded. After the initialization, K and A are processed block-by-block with making appropriate separation by XORing constant in capacity. To convert M to C, for each block, r-bit M_i is XORed to the state and the result is output as C_i. Then, the state is updated by sLiSCP permutation F. After all the ciphertext blocks are generated, the key K is absorbed to the state again, and the tag T is extracted from the state. The AE mode is described in Fig. 4 in Appendix.

Recommended Parameters and Security. The recommended parameters and security claims of the hashing mode and the AE mode are presented in Tables 3 and 4, respectively.

Table 3. Recommended parameters and bit securities in hashing mode

Algorithm	IV	Digest	r	r'	c	Collision
sLiSCP-192	0x502020	160	32	32	160	80
sLiSCP-256	0x604040	192	64	64	192	96
sLiSCP-256	0x604020	192	64	32	192	96

Table 4. Recommended parameters and bit securities in AE mode

Algorithm	Key	Nonce	Tag	r	c	Confidentiality	Integrity
sLiSCP-192/80	80	80	80	32	160	80	80
sLiSCP-192/112	112	80	112	32	160	112	112
sLiSCP-256/128	128	128	128	64	192	128	128

3 Differential Trail Search on sLiSCP

The core of our attacks is to find good differential trails. While there are many existing results on automated differential trail search tool, it is infeasible to apply those to sLiSCP permutations owing to their large state size and complicated step function using Simeck. In this section, we introduce our strategy to reduce the search problem for the entire permutation to several iterations of Simeck.

The search strategy depends on which of the permutation or the sponge mode is attacked.

Permutation: The number of attacked steps is large (i.e. 15 for our attacks), thus we search for an iterative differential trail for a small number of steps and iterate it several times. As it will be explained later, the rebound attack often utilizes sparse differential trails for an outbound phase, thus it is desired to start and end the iterative trail with a sparse difference.

Sponge mode: Considering that differences can be injected only through the message input to r bits of the state, the differential trail must start from and end with r-bit rate specified in Sect. 2. Hence, this is another iterative differential trail in a branch-wise level.

In the sponge mode, a half of the rate exists in the left half of the state, e.g. B_6, B_7 for sLiSCP-192, and the other half exists in the right half, e.g. B_{18}, B_{19}. We found that injecting differences in both halves decreases the probability quickly especially to satisfy the constraint that the output difference can only exist in r-bit rate. In the end, for both targets, our goal is to find an iterative difference that starts and ends with single active branch denoted by $(0, 0, 0, \alpha)$. Such trail can be found for 6 steps. Its schematic diagram is shown in Fig. 6 in Appendix A.

Maximizing the Search Space. By considering the attacks on the sponge mode, α in the output difference can be replaced with another one denoted by γ, which relaxes the constraint and may increases the probability of the trail. We then found that by fixing the differential propagation in the first and the last steps to $\alpha \rightarrow \beta$ and $\gamma \rightarrow \delta$, all the internal state differences are fixed, i.e. the search space is maximized. To attack the permutation, α and γ can take any m-bit difference, while to attack the sponge-mode, only $r/2$-bits can have differences. To discuss the differential trail on the sponge mode, it is convenient to denote the state (X_0, X_1, X_2, X_3) by using 8 $m/2$-bit words S_i as $(S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7)$, then the difference can only be injected to S_2 and S_6 in the sponge-based mode. The 6-step trail in the word-wise level is shown in Fig. 2. A coloured box indicates the propagation of nonzero differences.

The probability of the trail is

$$\Pr(\alpha \rightarrow \beta)^2 \times \Pr(\beta \rightarrow \gamma) \times \Pr(\gamma \rightarrow \delta)^2 \times \Pr(\delta \rightarrow \alpha).$$

As a consequence, we reduce the search problem for the entire sLiSCP permutation into the problem of 4 parallel searches on 6-round Simeck-48 or 8-round

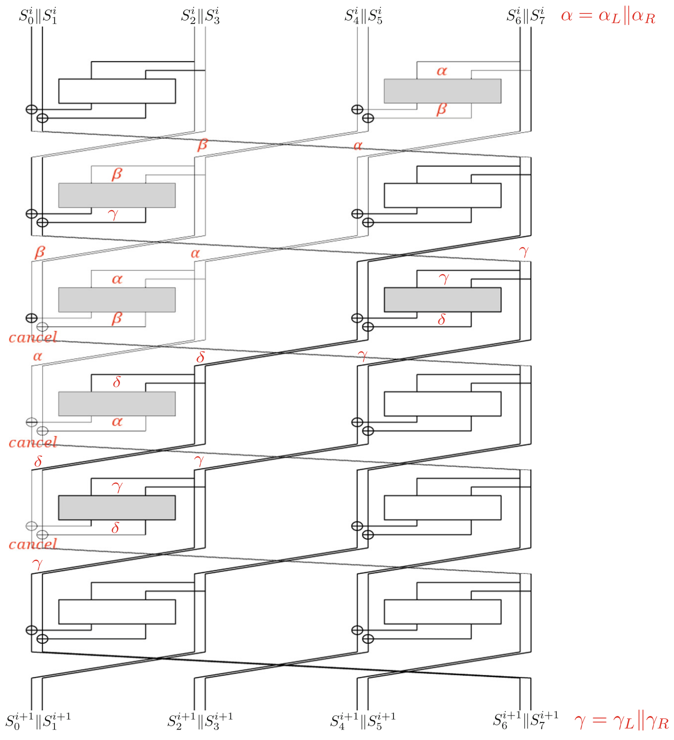


Fig. 2. 6-step differential trail for `sLiSCP`

Simeck-64, which seems feasible but requires clever coding to find the best combination of the results from the 4 parts. Interestingly, those 4 propagations form a circulation $\alpha \rightarrow \beta \rightarrow \gamma \rightarrow \delta \rightarrow \alpha$. Then, 4 differential propagations can be searched in a sequential way by regarding those 4 propagations as an iterative differential trail against 24-round Simeck-48 and 32-round Simeck-64, which is now feasible and easy to optimize the combined results with existing automatic search tools. We follow the automatic search model of Simon and Speck [22,24], due to their similar structures with Simeck. In other word, we have taken some dependencies in the round functions of Simeck into account in the automatic search. In addition, we experimentally verified the probabilities of some characteristics in 6/8-round Simeck found by the tool, and the results match the theoretical predictions of the differential probability. As an example, we show the detail of a 6-round differential trail for Simeck-48 in Table 10 in Appendix.

Search Results. Table 5 shows an overview of the distinguishers we found for 6-step `sLiSCP`. The differences α and γ are the input and output differences in the trails, which also define the differential for `sLiSCP`. The reference shows the applications of the distinguishers in this paper.

Table 5. An overview of the distinguishers found for sLiSCP-256 and sLiSCP-192

ID	Version	#steps	α	γ	Pr	ref.
Ω_1	sLiSCP-192	6	010000‖000000	010000‖000000	$2^{-103.96}$	Sect. 4,5
Ω_2	sLiSCP-256	6	08800000‖00000000	08800000‖00000000	$2^{-112.14}$	Sect. 4,5
Ω_3	sLiSCP-192	6	014000‖020000	014000‖020000	$2^{-88.8}$	Sect. 6
Ω_4	sLiSCP-256	6	00000000‖80000000	00000000‖80000000	$2^{-112.14}$	Sect. 6

The input differences of the trails Ω_1 and Ω_2 satisfy the restrictions from the sponge mode, while no such restrictions were considered in Ω_3 and Ω_4 towards the analysis on the permutation. The differential trails of Simeck^6-48 in the differential Ω_1 of sLiSCP-192 are shown as follows.

$$\alpha = \alpha_L \| \alpha_R \triangleq \texttt{010000} \| \texttt{000000} = \gamma, \quad \Pr[\alpha \xrightarrow{6R} \beta] = 2^{-20},$$
$$\beta \triangleq \texttt{1d0000} \| \texttt{060000} = \delta, \quad \Pr[\beta \xrightarrow{6R} \alpha] = 2^{-18},$$

In the following, we have the differential trails of Simeck^8-64 in sLiSCP-256.

$$\alpha = \alpha_L \| \alpha_R \triangleq \texttt{08800000} \| \texttt{00000000} = \gamma, \quad \Pr[\alpha \xrightarrow{8R} \beta] = 2^{-22},$$
$$\beta \triangleq \texttt{00800000} \| \texttt{00000000} = \delta, \quad \Pr[\beta \xrightarrow{8R} \alpha] = 2^{-22}.$$

Without the difference restriction, the probability of the trails can be improved for Simeck-48, such as Ω_3 of sLiSCP-192, which is shown below.

$$\alpha = \alpha_L \| \alpha_R \triangleq \texttt{014000} \| \texttt{020000} = \gamma, \quad \Pr[\alpha \xrightarrow{6R} \beta] = 2^{-12},$$
$$\beta \triangleq \texttt{014000} \| \texttt{008000} = \delta, \quad \Pr[\beta \xrightarrow{6R} \alpha] = 2^{-26}.$$

As for sLiSCP-256, even though there exist 6-step trails with larger probability than the optimal trail in Ω_2, the 6-step distinguisher Ω_2 has an overall advantage by taking the differential effect into account. Yet we still found a new distinguisher shown below Ω_4, which is similar to Ω_2. Considering the lower Hamming weight of α than Ω_2, this is more suitable for rebound attacks.

$$\alpha = \alpha_L \| \alpha_R \triangleq \texttt{00000000} \| \texttt{80000000} = \gamma, \quad \Pr[\alpha \xrightarrow{8R} \beta] = 2^{-22},$$
$$\beta \triangleq \texttt{00000000} \| \texttt{80000008} = \delta, \quad \Pr[\beta \xrightarrow{8R} \alpha] = 2^{-22}.$$

Interesting, we did not find any case that using different α and γ increases the probability. (In general, it occurs e.g. we confirmed the increase of the probability for different rate positions.) To make the paper simple, hereafter we use α and β instead of γ and δ, respectively.

Furthermore, the probability evaluation in Table 5 takes into account the effect of the differentials within Simeck. Specifically, we enumerate the trails

Table 6. The distribution of trails in the differential ($\texttt{014000}\|\texttt{020000} \xrightarrow{6R} \texttt{014000}\|\texttt{008000}$) of Simeck-48. The differential probability is approximately $2^{-11.3}$.

$-\log(p)$	12	13	14	15	16	17	18	19	20
#char	1	0	2	1	0	0	0	0	0

Table 7. The distribution of trails in the differential ($\texttt{014000}\|\texttt{008000} \xrightarrow{6R} \texttt{014000}\|\texttt{020000}$) of Simeck-48. The differential probability is approximately $2^{-21.8}$.

$-\log(p)$	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
#char	3	0	13	14	35	59	102	168	255	452	675	1021	1454	1907	2454	3081	3608

$-\log(p)$	43	44	45	46	47	48	49	50	51	52	53	54
#char	4141	4219	3859	3154	2280	1425	754	333	122	36	8	1

within the differentials of Simeck, which leads to a refined estimation of the probabilities for the distinguishers in sLiSCP. the distribution of trails according to the probability in ($\texttt{014000}\|\texttt{020000} \xrightarrow{6R} \texttt{014000}\|\texttt{008000}$) is shown in Table 6.

The differential probability is $2^{-11.3}$. Similarly in Table 7, we show the trails in the differential ($\texttt{014000}\|\texttt{008000} \xrightarrow{6R} \texttt{014000}\|\texttt{020000}$), where the probability is $2^{-21.8}$. With the differential effect taken into account, the probability of Ω_3 is approximately $2^{-88.8}$. For simplicity, we omitted the details of the trails in other distinguishers, and summarise the probabilities of the differentials in Simeck in Table 8.

Table 8. An overview for the probabilities of the obtained differentials

Version	Differential	Probability
Simeck^6-48	010000‖000000 → 1d0000‖060000	$2^{-17.85}$
	1d0000‖060000 → 010000‖000000	$2^{-16.28}$
Simeck^6-48	014000‖020000 → 014000‖008000	$2^{-11.3}$
	014000‖008000 → 014000‖020000	$2^{-21.8}$
Simeck^8-64	08800000‖00000000 → 00800000‖00000000	$2^{-18.69}$
	00800000‖00000000 → 08800000‖00000000	$2^{-18.69}$
Simeck^8-64	00000000‖80000000 → 00000000‖80000008	$2^{-18.69}$
	00000000‖80000008 → 00000000‖80000000	$2^{-18.69}$

4 6-Steps Forgery in AE Mode

In this section, the differentials explained in Sect. 3 are exploited for a forgery attack against 6-steps sLiSCP-192/112 and sLiSCP-256/128 in the AE mode. We apply the approach called "LOCAL attack" that was proposed by Khovratovich and Rechberger [21] and independently found by Wu et al. [29] against ALE [13].

4.1 Forgery

Let '$\|$' denote a concatenation. The attacker first observes a ciphertext having at least two encrypted message blocks $C_0\|C_1$. The ciphertext has a form $(N, A, C_0\|C_1, T)$, where N is a nonce, A is an associated data and T is a tag.

The attacker injects the difference specified in Sect. 3 to C^0 and C^1, namely $\overline{C^0} = C^0 \oplus (0\|\alpha_L)$ and $\overline{C^1} = C^1 \oplus (0\|\alpha_L)$. During the decryption, the difference injected by $\overline{C^0}$ makes the difference of S_6 to be α_L and this propagates through 6 steps so that it can be canceled by the difference from $\overline{C^1}$ with probability $2^{-103.96}$ for `sLiSCP`-192/112 and $2^{-112.14}$ for `sLiSCP`-256/128 (See Table 5). Hence the attacker makes decryption queries $(N, A, \overline{C^0}\|\overline{C^1}, T)$, which pass with the above probabilities.

The complexity of the attack against `sLiSCP`-192/112 is either $2^{103.96}$ data and $2^{103.96}$ verification attempts to achieve high success probability or 1 data and 1 verification attempts to achieve success probability of $2^{-103.96}$. The same applies to `sLiSCP`-256/128 by replacing $2^{103.96}$ with $2^{112.14}$.

4.2 Extension to State Recovery and Plaintext Recovery

In the duplex AE, the internal state value is always partially leaked as a ciphertext. Along with the information that a pair $C^0\|C^1$ and $\overline{C^0}\|\overline{C^1}$ satisfies the differential propagation, the attacker can recover the internal state as long as the number of candidate values of the internal state is sufficiently reduced. We show that the state recovery attack can be applied to `sLiSCP`-192.

We enumerate all the solutions of the first 4 active Simeck functions in Fig. 2. The differential for the first step is $\alpha \rightarrow \beta$ that is satisfied with probability $2^{-17.85}$. By examining all 2^{48} input values, $2^{48-17.85} = 2^{30.15}$ solutions will be found. We then further check the match with 24-bits of S_6 leaked by the key stream. $2^{30.15-24} = 2^{6.15}$ values match the observed key stream. In other words, the possible values of the 48-bit word $S_6\|S_7$ is now reduced to $2^{6.15}$ choices.

Similarly, the differential for the active Simeck function in the second step is $\beta \rightarrow \alpha$ that is satisfied with probability $2^{-16.28}$, the differential for the right function in step 3 is $\alpha \rightarrow \beta$ and the left function in step 3 is $\alpha \rightarrow \beta$ both are satisfied with probability $2^{-17.85}$. Hence, once the differential is satisfied, the number of possible state values for those Simeck^6-48 is $2^{31.72}$, $2^{30.15}$ and $2^{30.15}$ respectively. For any combination of paired values of those 4 Simeck functions, the 192-bit state values is uniquely fixed. In other words, the possible choices of the 192-bit state value are limited to the combination of those 4 Simeck functions. Hence the number of possible 192-bit states is $2^{6.15+31.72+30.15+30.15} = 2^{98.17}$.

Suppose that in the forgery attack, the encrypted message blocks is at least 6 blocks, and thus we make $\mathcal{D}(N, A, \overline{C^0}\|\overline{C^1}\|C_2\|C_3\|C_4\|C_5, T)$ in the forgery attack. Then the 128-bit value $C_2\|C_3\|C_4\|C_5$ can be used to filter out wrong candidates of $2^{98.17}$ choices of the 192-bit internal state.

In the end, for the state recovery, the data complexity increases to $6/2 \cdot 2^{103.96} = 2^{105.54}$. The computational complexity is $2^{105.54}$ memory access and $2^{98.17}$ 6-step `sLiSCP`-192 operations.

5 6-Steps Collision Attacks in Hashing Mode

We again use the 6-steps differential trail in Fig. 2. The forgery attacks in Sect. 4 are rather straightforward applications of the detected differentials. However, in the hash setting, the claimed bit-security is smaller, i.e. 80 bits (resp. 96 bits) for `sLiSCP`-192 (resp. `sLiSCP`-256), thus the naive approach with complexity $2^{103.96}$ (resp. $2^{112.14}$) is worse than the brute-force attack.

In the hash setting, attackers have access to the internal state value and can choose message values to control the differential propagation. This allows attackers to find collisions faster than the claimed bit-security for 6 steps.

5.1 Overall: Four-Block Collision Strategy

Our attacks find four-block colliding messages, namely $M^0\|M^1\|M^2\|M^3$ and $M^0\|M^1\|(M^2 \oplus 0\|\alpha_L)\|(M^3 \oplus 0\|\alpha_L)$ that produce the same hash digest.

No message difference is injected in the first and second message block. The purpose of those blocks is to set the state value that is advantageous to satisfy the 6-step differential trail in the third block. In short, the attacker precomputes all paired values that satisfy the differential propagation $\alpha \rightarrow \beta$ in the first step in Fig. 2 and $\beta \rightarrow \alpha$ in the second step. This allows the attackers to search for $M^0\|M^1$ producing the good values for the internal state after 2 blocks, denoted by $S_0^2\|S_1^2\|\cdots\|S_7^2$. Note that the reason why we need 2 blocks rather than 1 block is that degrees of freedom of a single message block, 2^{32} for `sLiSCP`-192 and 2^{64} for `sLiSCP`-256, are too small to find a colliding message pair.

The third block propagates differences as shown in Fig. 2 so that the output difference from the third block can be canceled out by injecting another message difference from the fourth message block.

5.2 Attack Procedure for `sLiSCP-256`

We first explain the attack for `sLiSCP`-256 that is instantiated with Simeck^8-64. We denote the left and right functions in step i, where $i \in \{0, 1, \cdots, 5\}$, by Simeck^8-64^{iL} and Simeck^8-64^{iR}, respectively. The illustration of the attack is shown in Fig. 7 in Appendix.

Precomputation

- For all $x_0 \in \{0,1\}^{64}$, compute Simeck^8-64$^{0R}(x_0) \oplus$ Simeck^8-64$^{0R}(x_0 \oplus \alpha)$ to check if the result is β or not. Because $\Pr[\alpha \xrightarrow{8R} \beta] = 2^{-18.69}$, we have $2^{64-18.69} = 2^{45.31}$ choices of x_0. Let y_0 be the corresponding output value for x_0. Those $2^{45.31}$ choices of (x_0, y_0) are stored in a table T^{0R}.

Let x_0^L and x_0^R be the left and right halves of x_0, namely $x_0 = x_0^L \| x_0^R$. T^{0R} is further sorted with respect to the 32-bit value of x_0^R. Because we have $2^{45.31}$ choices in T^{0R}, we expect $2^{45.31-32} = 2^{13.31}$ choices of x_0^L for each x_0^R.
In the end, a table T^{0R} of size $2^{45.31}$ is divided into 2^{32} tables $T_i^{0R}, i = 0, 1, \cdots, 2^{32}-1$, of size $2^{13.31}$ that store $2^{13.31}$ values of x_0^L for $x_0^R = i$.

– Do the same for Simeck^8-64^{1L}. Namely, for all $x_1 \in \{0,1\}^{64}$, compute Simeck^8-64$^{1L}(x_1) \oplus$ Simeck^8-64$^{1L}(x_1 \oplus \beta)$ to check if the result is α or not. Because $\Pr[\beta \xrightarrow{8R} \alpha] = 2^{-18.69}$, we have $2^{64-18.69} = 2^{45.31}$ choices of x_1. Let y_1 be the corresponding output value for x_1. Those $2^{45.31}$ choices of (x_1, y_1) are stored in a table T^{1L}.

The First Two Steps of the Differential. Choose $M_0 \| M_1$ uniformly at random and compute the second block output $S_0^2 \| S_1^2 \| \cdots \| S_7^2$. Thanks to the precomputation of Simeck^8-64^{0R}, for a given S_7^2, there are $2^{13.31}$ choices of x_0^L such that $x_0^L \| S_7^2$ satisfies the differential propagation $\alpha \rightarrow \beta$ for the first step. Moreover, the corresponding output y_0 is already stored in the table.

Hence, for a given S_4^2, S_5^2, S_7^2 and $2^{13.31}$ choices of y_0, compute $(S_4^2 \| S_5^2) \oplus y_0$ and check if this matches x_1 in the table T^{1L}. Considering that $2^{45.31}$ choices of x_1 are stored in T^{1L}, the probability of the match after $2^{13.31}$ iterations of y_0 is $2^{-64+45.31+13.31} = 2^{-5.38}$. Therefore, by choosing $2^{5.38}$ choices of $M_0 \| M_1$, we can find $M_0 \| M_1$ and $M_2^R \leftarrow x_0^L \oplus S_6^2$ such that the differential propagation for the first two steps are satisfied.

The Last Four Steps of the Differential. The attacker then uses the 32-bit value of M_2^L as degrees of freedom to satisfy the remaining 4 steps. The probability for the 4 steps is $2^{-18.69 \times 4} = 2^{-74.76}$. After examining 2^{32} choices of M_2^L, all the propagations are satisfied with probability $2^{-74.76+32} = 2^{-42.76}$.

Hence, by iterating the attack procedure so far $2^{42.76}$ times, the attacker can find a desired message pair $M_0 \| M_1 \| M_2$ and $M_0 \| M_1 \| (M_2 \oplus 0 \| \alpha_L)$. Then, the output difference from the third block can be easily canceled by the message difference for the fourth block.

Complexity Analysis. Complexity of the precomputation phase is $2 \cdot 2^{64} = 2^{65}$. It requires a memory to store $2 \cdot 2^{45.31} = 2^{46.31}$ values. The complexity to satisfy the 6-step differential up to the first two steps is $2^{5.38+13.31} = 2^{18.69}$. The complexity to satisfy all the 6-step differential is $2^{65} + 2^{42.76}(2^{18.69} + 2^{32}) \approx 2^{74.76}$. This is faster than the generic attack complexity of 2^{96}.

5.3 Attack Procedure for sLiSCP-192

The attack for sLiSCP-192 is basically the same as one for Simeck-64. The only differences are the state size and the probability of the differentials. We briefly explain the attack for sLiSCP-192.

Precomputation

– Examine $x_0 \in \{0,1\}^{48}$ input values to Simeck^6-48^{0R} to pick up all values satisfying the differential propagation $\alpha \xrightarrow{6R} \beta$ that can be satisfied with probability $2^{-17.85}$. As a result, $2^{30.15}$ choices of (x_0, y_0) are stored in a table T^{0R}, and there are about $2^{30.15-24} = 2^{6.15}$ choices of x_0^L for each of x_0^R.
– For Simeck^6-48^{1L}, the probability of the differential $\beta \xrightarrow{6R} \alpha$ is $2^{-16.28}$. We obtain $2^{31.72}$ choices of x_1 satisfying this differential propagation for T^{1L}.

The First Two Steps of the Differential. Choose $M_0\|M_1$ and the corresponding $S_0^2\|S_1^2\|\cdots\|S_7^2$. For a given S_7^2, there are $2^{6.15}$ choices of x_0^L and the corresponding y_0. Then, for a given $S_4^2\|S_5^2, S_7^2$ and $2^{6.15}$ choices of y_0, compute $S_4^2, S_5^2 \oplus y_0$ and check if this matches x_1. Considering that $2^{31.72}$ choices of x_1 are stored, the probability of the match after $2^{6.15}$ iterations of y_0 is $2^{-48+31.72+6.15} = 2^{-10.13}$. Therefore, by choosing $2^{10.13}$ choices of $M_0\|M_1$, we find $M_0\|M_1$ and M_2^R satisfying the differential trail for the first two steps.

The Last Four Steps of the Differential. The attacker uses 24-bit values of M_2^L as degrees of freedom to satisfy the remaining 4 steps. The probability for the 4 steps is $2^{-17.85*3-16.28} = 2^{-69.83}$. After examining 2^{24} choices of M_2^L, the probability of the remaining 4 steps is $2^{-69.83+24} = 2^{-45.83}$. Hence, by iterating the attack procedure so far $2^{45.83}$ times, a collision is generated.

Complexity Analysis. Complexity of the precomputation phase is $2 \cdot 2^{48} = 2^{49}$. It requires a memory to store $2^{30.15} + 2^{31.72} = 2^{32.14}$ values. The complexity to satisfy the 6-step differential up to the first two steps is $2^{10.13+6.15} = 2^{16.28}$. The complexity to satisfy all the 6-step differential is $2^{49} + 2^{45.83}(2^{16.28} + 2^{24}) \approx 2^{69.83}$. This is faster than the generic attack complexity of 2^{80}.

Table 9. Configuration for rebound attacks

Steps (i)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Propagation in Simeck^{iR}	0	0	I	0	I	0	0	0	I	0	I	0	0	0	(I)
Propagation in Simeck^{iL}	I	0	0	II	I	II	I	0	0	II	I	II	I	0	0
Configuration		F_b			F_{in}						F_f				

'I' and 'II' denote the differential trail $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$, respectively. "(I)" in step 14 denotes that the attacker accepts any output difference from this Simeck function without paying any cost. For sLiSCP-192, 'I' and 'II' are satisfied with probability $2^{-11.3}$ and $2^{-21.8}$, respectively. For sLiSCP-256, both are satisfied with probability $2^{-18.7}$.

6 15-Steps Rebound Attacks Against sLiSCP Permutation

Because sLiSCP is a cryptographic permutation, we also discuss its security as a permutation. We apply the rebound attack [23,25] to show that the differential-based approach can detect non-ideal behaviours for a large number of steps.

Goal of Rebound Attacks. Let x_i and y_i be an input and output of the sLiSCP permutation, respectively, namely $y_i = \texttt{sLiSCP}(x_i)$. The goal of the rebound attack is to find (x_1, y_1) and (x_2, y_2) where $x_1 \oplus x_2$ and $y_1 \oplus y_2$ belong to a predefined input subspace and output subspace, respectively. If an attacker can find such (x_1, y_1) and (x_2, y_2) against the target permutation faster than a random permutation, the target construction is regarded as non-ideal. This framework is called *limited-birthday distinguisher* (LBD) [16].

The generic attack complexity of LBD was proven by Iwamoto et al. [20]. Let $\mathcal{X}$ and $\mathcal{Y}$ be closed sets of input and output differences. Let also n be a permutation size. Then the generic attack complexity to solve LBD is

$$2^{n+1}/|\mathcal{X}| \cdot |\mathcal{Y}|. \tag{1}$$

An attacker builds a differential trail and divides the target permutation F into three consecutive parts F_b, F_{in}, and F_f, that is, $F = F_f \circ F_{in} \circ F_b$. The attacker first enumerates all the paired values satisfying the differential trail for F_{in}. This is called *an inbound phase* and the collected solutions are called *starting points*. Then the attacker propagates each starting point to F_f and F_b to probabilistically satisfy the differential trails. This is called *an outbound phase*, which is a brute force search by using starting points as degrees of freedom.

Overall Strategy. The most important part of the rebound attack is searching for efficient differentials. We use the 6-step differentials shown in Fig. 2 that was designed to be iterated multiple times. Because the analysis target is a permutation, we do not have to consider the limitation from the message injection positions in the sLiSCP mode. Thus we use the differentials Ω_3 in Table 5 for sLiSCP-192 and Ω_4 for sLiSCP-256.

The distribution of active Simeck functions for 15 steps is shown in Table 9. As in Table 9 and Fig. 8 in Appendix, we locate the inbound phase from steps 4 to 6. This is because if we fix values for 4 active Simeck functions, the entire state value will be fixed. We choose 4 active Simeck functions to cover the lowest probability part, so that the probability of the outbound phase is maximized.

6.1 Attack Procedure for sLiSCP-192

Inbound Phase. We first enumerate all the solutions for the active Simeck-48 functions in the inbound phase. For example, in Step 3, for all $x \in \{0,1\}^{48}$, compute Simeck^6-$48^{3L}(x) \oplus$ Simeck^6-$48^{3L}(x \oplus \beta)$ matches the output difference α. If so, we store the solutions in a table T^{3L}. Because the probability of the differential $\beta \overset{6steps}{\longrightarrow} \alpha$ is $2^{-21.8}$, we expect $2^{48-21.8} = 2^{26.2}$ solutions.

Apply the same procedure for Simeck^6-48^{4L}, Simeck^6-48^{4R}, Simeck^6-48^{5L} to store the solutions to tables T^{4L}, T^{4R}, and T^{5L}. Considering that the probability of the differential $\alpha \overset{6steps}{\longrightarrow} \beta$ is $2^{-11.3}$, we expect $2^{48-11.3} = 2^{36.7}$ solutions for T^{4L}, $2^{36.7}$ solutions for T^{4R} and $2^{26.2}$ solutions for T^{5L}.

Outbound Phase. If we fix one solution for each of four active Simeck functions in the inbound phase, the entire 192-bit state value is uniquely fixed. Hence, we propagate the values to F_b and F_f to check if the outbound phase is satisfied.

The number of total starting points is $2^{(2\times 36.7)\times(2\times 26.2)} = 2^{125.8}$, while the probability for F_f is $2^{(6\times -11.3)\times(2\times -21.8)} = 2^{-111.4}$, and the probability for the F_b is $2^{2\times -11.3} = 2^{-22.6}$, in which the total probability is 2^{-134}. Hence, the degrees of freedom is not sufficient to fully satisfy the 15-step differentials

$$(\beta, \alpha, 0, 0) \overset{15steps}{\longrightarrow} (0, \beta, \alpha, 0).$$

Hence, we relax the differential and accept any 48-bit difference in the second word of the output difference, namely,

$$(\beta, \alpha, 0, 0) \overset{15steps}{\longrightarrow} (0, *, \alpha, 0).$$

This increases the probability of the outbound phase to $2^{-134+11.3} = 2^{-122.7}$, which can be satisfied with $2^{125.8}$ starting points.

Complexity Evaluation. The inbound phase requires $4 \cdot 2^{48} = 2^{50}$ computations and a memory to store $2^{26.2} + 2^{36.7} + 2^{36.7} + 2^{26.2}$ words for T^{3R}, T^{4L}, T^{4R}, and T^{5L}, which is about $2^{37.7}$ words. The outbound phase requires $2^{122.7}$ computations to satisfy the differential propagations. In the end, the complexity of the attack is $2^{122.7}$ computations and $2^{37.7}$ memory amount.

The complexity to find the same paired values in a random function is much higher. Indeed, the subspace of the input difference is fixed to one choice, thus $|\mathcal{X}| = 1$. The subspace of the output difference is fixed but for the second word, thus $|\mathcal{Y}| = 2^{48}$. From Eq. (1), the generic attack complexity is $2^{192+1}/(1 \cdot 2^{48}) = 2^{145}$, which is higher than our rebound attack complexity.

6.2 Attack Procedure for `sLiSCP-256`

We again divide the target 15-steps as shown in Table 9. The evaluation for `sLiSCP-256` is much simpler than the case of `sLiSCP-192` because the probabilities of the both differentials $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$ are $2^{-18.7}$.

In the inbound phase, we enumerate all the solutions of four active Simeck-64 functions. We obtain $2^{64-18.7} = 2^{45.7}$ solutions for each that are stored in four tables of size $2^{45.7}$. Considering all the combination of the solutions, we can generate up to $2^{4\times 45.7} = 2^{182.8}$ starting points.

In the outbound phase, F_b and F_f contain 2 and 8 active Simeck-64 functions, respectively, thus the entire probability is $2^{10\times -18.7} = 2^{-187.0}$. Here, we again accept any output difference in the last step, which increases the probability to $2^{9\times -18.7} = 2^{-168.3}$ and makes the input and output differences of the 15-step `sLiSCP`-256 as $(\beta, \alpha, 0, 0)$ and $(0, *, \alpha, 0)$.

The complexity of the rebound attack is $2^{168.3}$ computations and memory to store $4 \cdot 2^{45.7} = 2^{47.7}$ values. The complexity to satisfy the same input and output differences against a random permutation is $2^{256+1}/(1 \cdot 2^{64}) = 2^{193}$, which is higher than our rebound attack.

7 Concluding Remarks

In this paper, we investigated the security of `sLiSCP` permutation, especially the first security analysis in the AE and hash settings defined as the sponge-based construction. We first explained our differential trail search strategy that reduces the search problem of the entire permutation to 24-round Simeck-48 and 32-round Simeck-64. This allowed us to run an existing tool. Based on the detected trail, we performed forgery and state-recovery for 6-steps AE, collision attacks on 6-steps hash and rebound distinguishers on 15-steps permutation. We believe that our several analyses respecting the constraints by the mode will provide a better understanding of the security of `sLiSCP`.

Acknowledgements. We thank the anonymous reviewers for their valuable comments. This work was initiated during the 7'th Asian Workshop on Symmetric Key Cryptography, we would like to thank the organisers of ASK 2017. Yunwen Liu is supported by the Research Fund KU Leuven C16/18/004, grant agreement No. H2020-MSCA-ITN-2014-643161 ECRYPT-NET, China Scholarship Council (CSC 201403170380) and National Natural Science Foundation (No. 61672530). Ling Song is supported by the Youth Innovation Promotion Association CAS and the National Natural Science Foundation of China (Grants No. 61802399, 61472415, 61732021 and 61772519). Gaoli Wang is supported by the National Natural Science Foundation of China (No. 61572125) and National Cryptography Development Fund (No. MMJJ20180201).

A Appendix

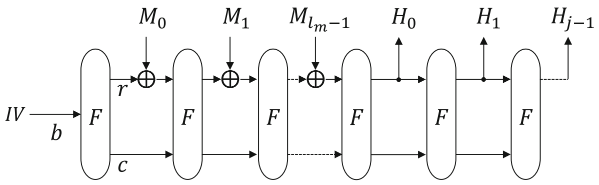


Fig. 3. `sLiSCP` hashing mode

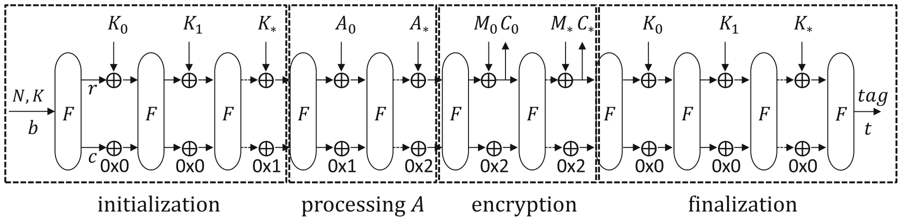


Fig. 4. `sLiSCP` AE mode

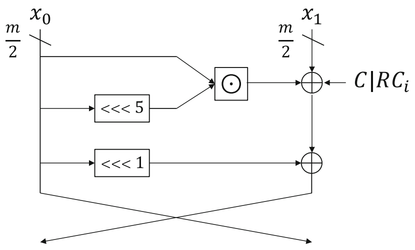


Fig. 5. 1-round of Simeck

Table 10. A 6-round differential trail of Simeck-48 with probability 2^{-12}.

Round	Left difference	Right difference	Probability
0	014000	020000	
1	008000	014000	2^{-4}
2	004000	008000	2^{-2}
3	000000	004000	2^{-2}
4	004000	000000	1
5	008000	004000	2^{-2}
6	014000	008000	2^{-2}
Total			2^{-12}

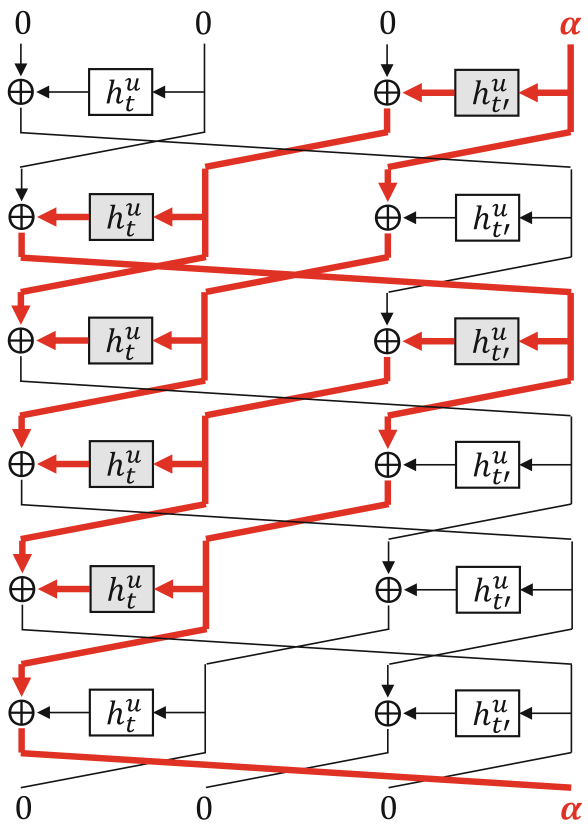


Fig. 6. 6-round iterative differential trail with single active branch. A coloured box indicates the propagation of nonzero differences. (Color figure online)

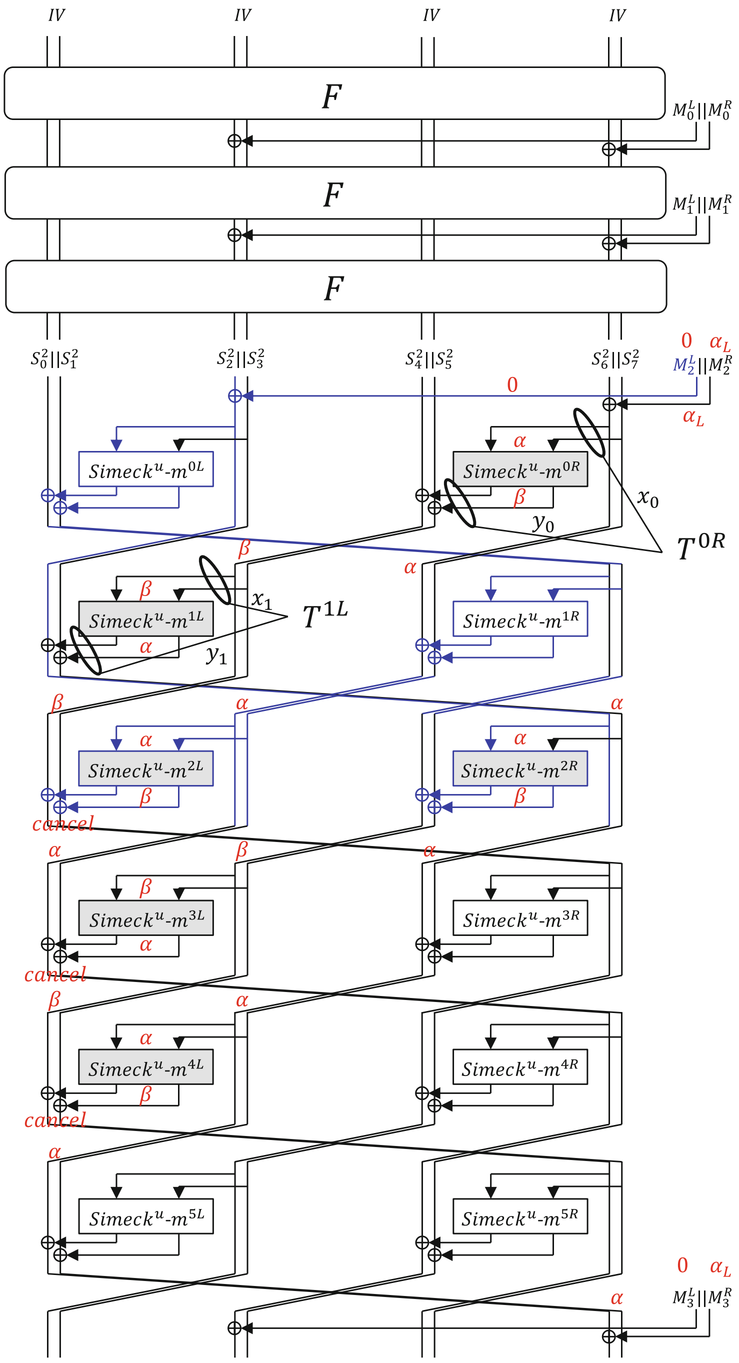
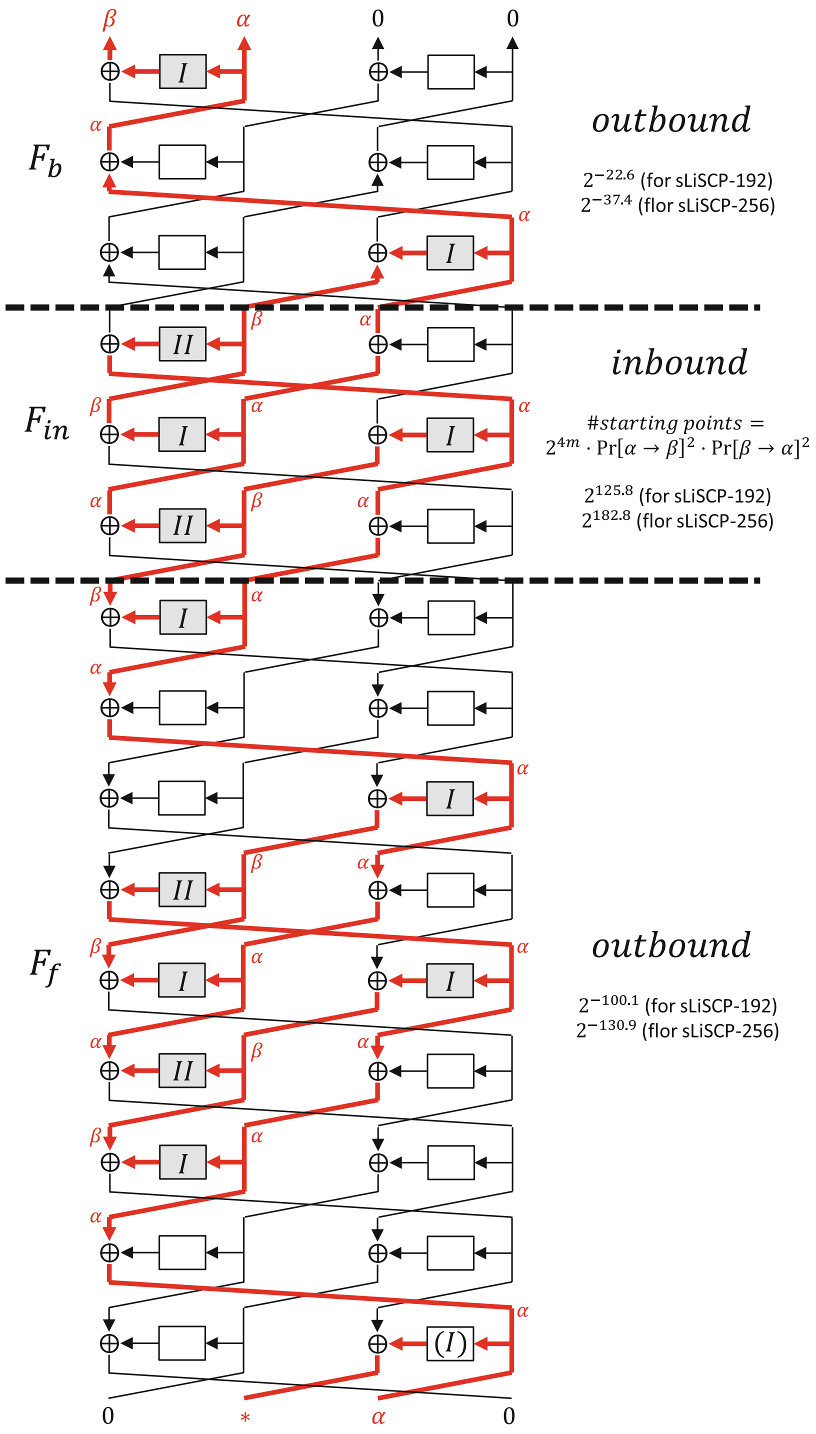


Fig. 7. 6-step collision attack. $\alpha = \alpha_L \| \alpha_R$, and α_R is set to 0. Blue lines show the impact of modifying M_2^L up to step 2, which does not impact to the active Simeck functions in steps 0 and 1, and impacts to all the Simeck functions in steps 2 to 5. (Color figure online)



β
α
0
0
I
F_b
outbound
$2^{-22.6}$ (for sLiSCP-192)
$2^{-37.4}$ (flor sLiSCP-256)
α
II
F_{in}
inbound
#starting points = $2^{4m} \cdot \Pr[\alpha \to \beta]^2 \cdot \Pr[\beta \to \alpha]^2$
$2^{125.8}$ (for sLiSCP-192)
$2^{182.8}$ (flor sLiSCP-256)
F_f
outbound
$2^{-100.1}$ (for sLiSCP-192)
$2^{-130.9}$ (flor sLiSCP-256)
(I)
0
*
α
0

Fig. 8. Differential trail for 15-step rebound attack (Color figure online)

References

1. AlTawy, R., Rohit, R., He, M., Mandal, K., Yang, G., Gong, G.: sLiSCP: simeck-based permutations for lightweight sponge cryptographic primitives. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 129–150. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72565-9_7
2. AlTawy, R., Rohit, R., He, M., Mandal, K., Yang, G., Gong, G.: sLiSCP-light: towards lighter sponge-specific cryptographic permutations (2018). https://cacr.uwaterloo.ca/techreports/2018/cacr2018-01.pdf
3. Aumasson, J., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: a lightweight hash. J. Cryptol. **26**(2), 313–339 (2013). https://doi.org/10.1007/s00145-012-9125-6
4. Aumasson, J.-P., Jovanovic, P., Neves, S.: NORX: parallel and scalable AEAD. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 19–36. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11212-1_2
5. Babbage, S., Dodd, M.: The MICKEY stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 191–209. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_15
6. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptology ePrint Archive 2013, 404 (2013). http://eprint.iacr.org/2013/404
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_11
8. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28496-0_19
9. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponge functions. Submission to NIST (Round 3) (2011). http://sponge.noekeon.org/CSF-0.1.pdf
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR submission: Ketje v2. Candidate of CAESAR Competition, September 2016
11. Bogdanov, A., Knezevic, M., Leander, G., Toz, D., Varici, K., Verbauwhede, I.: SPONGENT: a lightweight hash function. In: Preneel and Takagi [27], pp. 312–325. http://dx.doi.org/10.1007/978-3-642-23951-9_21
12. Bogdanov, A., et al.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_31
13. Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., Tischhauser, E.: ALE: AES-based lightweight authenticated encryption. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 447–466. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43933-3_23
14. De Cannière, C., Preneel, B.: TRIVIUM. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_18
15. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Submission to the CAESAR competition. Submission to NIST (Round 3) (2016). http://competitions.cr.yp.to/round3/asconv12.pdf

16. Gilbert, H., Peyrin, T.: Super-sbox cryptanalysis: improved attacks for AES-like permutations. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13858-4_21
17. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight hash functions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 222–239. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_13
18. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED block cipher. In: Preneel and Takagi [27], pp. 326–341. http://dx.doi.org/10.1007/978-3-642-23951-9
19. Hell, M., Johansson, T., Maximov, A., Meier, W.: The grain family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_14
20. Iwamoto, M., Peyrin, T., Sasaki, Y.: Limited-birthday distinguishers for hash functions. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 504–523. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_26
21. Khovratovich, D., Rechberger, C.: The LOCAL attack: cryptanalysis of the authenticated encryption scheme ALE. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 174–184. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_9
22. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 161–185. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_8
23. Lamberger, M., Mendel, F., Schläffer, M., Rechberger, C., Rijmen, V.: The rebound attack and subspace distinguishers: application to whirlpool. J. Cryptol. **28**(2), 257–296 (2015)
24. Liu, Y., De Witte, G., Ranea, A., Ashur, T.: Rotational-XOR cryptanalysis of reduced-round SPECK. IACR Trans. Symmetric Cryptol. **2017**(3), 24–36 (2017)
25. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: cryptanalysis of reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03317-9_16
26. NIST: Lightweight Cryptography, April 2018. https://csrc.nist.gov/projects/lightweight-cryptography
27. Preneel, B., Takagi, T. (eds.): CHES 2011. LNCS, vol. 6917. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9
28. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_12
29. Wu, S., Wu, H., Huang, T., Wang, M., Wu, W.: Leaked-state-forgery attack against the authenticated encryption algorithm ALE. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 377–404. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_20
30. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The Simeck family of lightweight block ciphers. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 307–329. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_16