Automatic Classical and Quantum Rebound Attacks on AES-like Hashing by Exploiting Related-key Differentials

Xiaoyang Dong[1], Zhiyu Zhang[3,4], Siwei Sun[2,5⋆], Congming Wei[1]
Xiaoyun Wang[1,6,7], and Lei Hu[3,4]

[1] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China.
`{xiaoyangdong,wcm16,xiaoyunwang}@tsinghua.edu.cn`
[2] School of Cryptology, University of Chinese Academy of Sciences, Beijing, China.
`siweisun.isaac@gmail.com`
[3] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.
`{zhangzhiyu,hulei}@iie.ac.cn`
[4] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China.
[5] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China.
[6] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, China.
[7] School of Cyber Science and Technology, Shandong University, Qingdao, China

Abstract. Collision attacks on AES-like hashing (hash functions constructed by plugging AES-like ciphers or permutations into the famous PGV modes or their variants) can be reduced to the problem of finding a pair of inputs respecting a differential of the underlying AES-like primitive whose input and output differences are the same. The rebound attack due to Mendel et al. is a powerful tool for achieving this goal, whose quantum version was first considered by Hosoyamada and Sasaki at EUROCRYPT 2020. In this work, we automate the process of searching for the configurations of rebound attacks by taking related-key differentials of the underlying block cipher into account with the MILP-based approach. In the quantum setting, our model guide the search towards characteristics that minimize the resources (e.g., QRAM) and complexities of the resulting rebound attacks. We apply our method to `Saturnin-hash`, `SKINNY`, and `Whirlpool` and improved results are obtained.

Keywords: Quantum computation · Collision attacks · Rebound attacks · `Saturnin` · `SKINNY` · `Whirlpool` · MILP

1 Introduction

A cryptographic hash function is a primitive that maps a binary string of arbitrary length into a short fixed-length digest, enjoying collision resistance, preim-

⋆ Corresponding author.

age resistance, and second-preimage resistance. One popular approach for building a cryptographic hash function is to plug a secure block cipher into one of the twelve secure PGV modes [45] to build the compression function, and then iterate it with the Merkle-Damgård paradigm [13,40]. In this work, we focus on the collision resistance of hash functions constructed in this way with AES-like ciphers (named as AES-like hashing) in both the classical and quantum setting.

The differential attack plays an important role in analyzing the collision resistance of a hash function H, since a successful collision attack implies a pair of inputs x and x' with nonzero difference $x \oplus x'$ such that the output difference $H(x) \oplus H(x')$ is zero. In the context of AES-like hashing, due to the feed-forward mechanism of the PGV modes, a collision means the identification of a pair of different inputs conforming a differential of the underlying block cipher whose input and output differences are the same. To be more concrete, let us consider the MMO mode (one of the twelve secure PGV modes) shown in Figure 1: $H(x) \oplus H(x') = 0$ implies $(m \oplus E_K(m)) \oplus (m \oplus \Delta \oplus E_K(m \oplus \Delta)) = 0$ or $E_K(m) \oplus E_K(m \oplus \Delta) = \Delta$. Therefore, finding a collision is equivalent to finding a pair conforming a differential of the underlying block cipher whose input and output differences are of the same value. One method for achieving this goal is the so-called rebound attack [38], which is the main technique involved in this work.

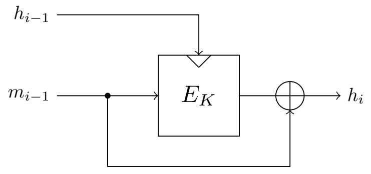


Fig. 1: (`MMO`) Matyas-Meyer-Oseas

1.1 The Rebound Attack

The rebound attack was first introduced by Mendel et al. at FSE 2009 [38]. Essentially, it is a technique for generating a pair of inputs fulfilling a differential $\delta \to \Delta$ for a block cipher. In the rebound attack, the targeted primitive with a truncated differential trail whose input and output differences share a common pattern is divided into three parts as shown in Figure 2. Then, the attacker generates a lot of pairs (named as starting points in the literature) conforming the inbound differential. Finally, the starting points are propagated forward and backward to identify data pairs fulfilling the outbound differentials and the additional constraint that the input and output differences of the whole trail should be equal.

To increase the number of rounds covered by the inbound differential for AES-like ciphers, the super S-box technique was introduced independently by Gilbert

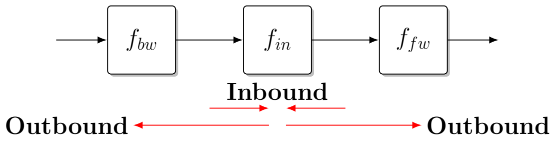


Fig. 2: The Rebound Attack

et al. [20] and Lamberger et al. [37], where two consecutive AES-like rounds are considered as a whole with several super S-boxes. Later, Sasaki et al. [47] showed that the memory complexity of the rebound attack can be significantly reduced by exploiting the differential property of non-full-active Super S-boxes. At CRYPTO 2011, Naya-Plasencia further improved the rebound attack by using better algorithms for merging large lists and finding solutions of the underlying differential trail [42]. The Rebound attack has become a basic technique for collision attacks [48,29,31,30,39,16] and distinguishing attacks on various hash functions. It even finds applications in the context of DS-MITM attacks [15,14].

The Role of the Key Expansions. In rebound attacks, the generation of the starting points relies on the degrees of freedom from the encryption data path of the underlying block cipher. A natural idea is to utilize the degrees of freedom from the key-schedule algorithm if we do not require the key to be a prefixed value (e.g., the IV). For the sake of simplicity, let us consider the MMO mode with a single message block (see Figure 1). A standard collision message pair (m, m') satisfies $H(IV, m) = H(IV, m')$, where the master key of the underlying block cipher is fixed and thus no degrees of freedom from the key-schedule algorithm can be used. However, for a semi-free-start collision $H(u, m) = H(u, m')$ $(u \neq IV)$ or a free-start collision $H(v, m) = H(v', m')$ $(v \neq v')$, the key is allowed to be changed and thus the degrees of freedom from the key-schedule algorithm may be utilized. At ASIACRYPT 2009, Lamberger et al. presented some improved semi-free-start collision attacks on reduced `Whirlpool` by exploiting the degrees of freedom from the key schedule algorithm [37]. Since there is no difference introduced into the key schedule data path, this type of attack can be modeled with the MILP-based method presented in [25,18]. At ASIACRYPT 2012, Sasaki et al. [48] applied the rebound attack on `Whirlpool` with an 8-round related-key truncated differential trail and find an 8-round free-start collision attack. To the best of our knowledge, no automatic method is available to find such free-start collisions based on the rebound attack. Finally, we would like to emphasize the importance of free-start collision attacks: The Merkle-Damgård security reduction assumes that any type of collision for the compression function should be intractable for the attacker, including free-start collisions.

1.2 Collision Attacks with Quantum Computing

For a long time, it was believed that quantum computing would have a limited impact on symmetric ciphers due to the quadratic speedup of an exhaustive search attack based on Grover's algorithm [24]. In ISIT 2010, Kuwakado and Morii showed how to break some provable secure schemes in the quantum setting [35], and this naive view started to change. Some follow-up works break more constructions [36,33]. However, a key step in these attacks involving the application of Simon's algorithm on a function with a hidden period related to the secret key, which requires the access to the keyed quantum oracle of the target. This is a strong requirement whose practical relevance is questioned. Hence, quantum attacks with higher complexities are still meaningful if they do not need to make online queries to superposition oracles of keyed primitives [7,28,34,43,23,27,6].

As keyless primitives, hash functions can be quantumly implemented offline and the thus attackers can freely make quantum superposition queries. For a hash function with n-bit output, classical algorithms find collisions with time complexity $O(2^{n/2})$. In the quantum setting, we have the following bounds induced by generic quantum attacks on hash functions.

- The BHT algorithm [8] equipped with a qRAM with size S finds a collision with a time complexity $T = \frac{2^{n/2}}{\sqrt{S}}$. It achieves optimal tradeoff when $T = 2^{n/3}$ and $S = 2^{n/3}$.
- Since the existence of large qRAM is still doubtful [22,21], there is a time-space tradeoff attack without qRAM, namely the quantum version of parallel rho's algorithm [49,25,4]. It achieves a time complexity of $T = \frac{2^{n/2}}{S}$ with S processors.
- The CNS algorithm [10] finds a collision with time complexity $T = 2^{2n/5}$ requiring a classical memory of size $2^{n/5}$ and $O(n)$ qubits.

At EUROCRYPT 2020, Hosoyamada and Sasaki [25] introduced the first dedicated quantum attack on hash functions (a quantum version of the rebound attack), which reveals that a differential trail whose probability is too low to be used in the classical setting may be exploitable in quantum attacks. However, the presented attacks are inferior to the CNS attack when there is no large qRAMs. At ASIACRYPT 2020, Dong et al. [18] reduced or even avoid the use of qRAM in the quantum rebound attacks by leveraging the non-full-active Super S-box technique. Recently, Hosoyamada and Sasaki [26] converted the classical semi-free-start collision attack on reduced SHA-2 into quantum collision attack and significantly improved the number of rounds attacked. At ToSC 2021, Chauhan et al. [11] found quantum collisions on reduced `AES-256` in double block length hashing. Ni et al. [44] investigated the quantum collision attacks on reduced `Simpira v2` in hashing modes.

1.3 Our Contribution

In this paper, we introduce an automatic tool to determine the related-key differentials, which are optimized for rebound attacks. More concretely, we focus on the free-start collision attacks based on rebound attack technique.

The main task is to increase the probability of the differential trail of the outbound part by properly consuming the degrees of freedom of the key. In addition, we have to deal with the *linear incompatibility*, which are frequently encountered in various automatic tools about related-key differential on AES-like ciphers, such as [5,19,12]. At CRYPTO 2013, Fouque et al. [19] find that the difference cancellation between the AES-128's key state and the round state in some round imposes some linear relationship between the key and state differences. Hence, difference cancellation in a different round cannot be independently simulated.

On ciphers with linear key schedule, Cid et al. [12] described an MILP model to search the related-key differentials, i.e., `Deoxys-BC` [32]. Since the relationship between `Deoxys-BC`'s round keys are somewhat weakened by the LFSRs, they do not need to consider incompatibilities between many rounds. In this paper, we study a more complex case, i.e., `Saturnin` [9], a round 2 candidate of NIST LWC competition, proposed by Canteaut et al. In `Saturnin`, the round keys are identical for the even or odd rounds, respectively, and the round key in the odd rounds are derived by shifting the key in even round by 5 cells. Hence, the relationships between the round keys in `Saturnin` are stronger, and Cid et al.'s model may lead to many incompatible solutions for `Saturnin`. To deal with the problem, we build an efficient method to fast abandon the incompatible solutions, where the incompatibilities come from many rounds, for example, contradictions between the truncated differentials in round 0 and round 6. In addition, we also model the inbound phase with key differences, where both the 2-round and 3-round inbound phases are considered. We build a uniform objective function on the time complexity to perform the rebound attack, that takes the complexity of solving the inbound phase and the probability of the outbound phase as a whole. Thereafter, we find an 8-round trail for the rebound attacks and generate an 8-round quantum free-start collision attack on the compression function of `Saturnin-hash`. In addition, we also identify a 7-round quantum collision attack on `Saturnin-hash` based on a 7-round single-key rebound attack trail.

We also apply the automatic model to `SKINNY`-128-384 [3]. Since `SKINNY` adopts non-MDS matrix, we build a dedicated method to solve the super S-box with non-MDS matrix. Compared to the usual super S-box with MDS matrix, our method explores the details of the non-MDS matrix of `SKINNY` and decomposes the super S-box into a sequence of small S-boxes. Our super S-box technique with non-MDS matrix does not need to precompute the differential distribution of the super S-box even in the full active case, which works efficiently in quantum attack without qRAM and large classic memory. Concretely, about $\sqrt{2^c}$ time is needed to solve the full active super S-box with non-MDS matrix quantumly without qRAM, while the time is $\sqrt{2^{dc}}$ for full active super S-box with MDS matrix, where $d = 4$ for `SKINNY` and `AES`. Thereafter, we give the 16-round free-start quantum collision attacks on the hashing modes with SKINNY-128-384.

On ciphers with nonlinear key schedule, we study the compression function of ISO standard hash function, Whirlpool [2]. In the automatic model, we place the 3-round inbound phase in both the key schedule path and data encryption path (we do not find better trail with the two-round inbound phases). In its quantum attack, we nest mutiple Grover's algorithms to solve several local searching problems. For Saturnin, the role of the consumption of degrees of freedom for key schedule is mainly to increase the probability of the outbound phase of the encryption data path. However, for Whirlpool, we have to consume the degrees of freedom of the key to increase the probabilities of the outbound phases in both the key schedule and the encryption data path. Finally, we introduce a 9-round quantum free-start collision attack on the compression function of Whirlpool, while the best previous attack is 8-round in classical setting [48]. The results are summarized in Table 1. Our quantum attacks do not need qRAM or classical memories, which perform better than the generic quantum collision attacks by parallel rho's algorithm [49,25,4]. However, the time complexities may be inferior to the quantum attacks equipped with large classical memory by Chailloux, Naya-Plasencia, and Schrottenloher's algorithm [10].

2 Preliminaries

In this section, we give a brief introduction of quantum computation, and familiarize the readers with the so-called super S-box and non-full-super S-box technique, etc. employed in our work.

2.1 Quantum Computation and Quantum RAM

The state space of an n-qubit quantum system is the set of all unit vectors in $\mathbb{C}^{2^n}$ under the orthonormal basis $\{|0\cdots 00\rangle, |0\cdots 01\rangle, \cdots, |1\cdots 11\rangle\}$, alternatively written as $\{|i\rangle : 0 \leq i < 2^n\}$. Quantum computation is achieved by manipulating the state of an n-qubit system by a sequence of unitary transformations and measurements.

Superposition Oracles for Classical Circuit. The superposition oracle of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is the unitary transformation $\mathcal{U}_f$ acting on an $(n+1)$-qubit system with the following functionality

$$\mathcal{U}_f\left(\sum_{x\in\mathbb{F}_2^n} a_i |x\rangle |y\rangle\right) = \sum_{x\in\mathbb{F}_2^n} a_i |x\rangle |y \oplus f(x)\rangle .$$

Grover's Algorithm. Given a quantum black-box access to a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ with $0 < f^{-1}(1) \ll 2^n$. Grover's algorithm finds an element $x \in \mathbf{F}_2^n$

Table 1: A Summary of the results.

Whirlpool							
Target	Attack	Rounds	Time	C-Mem	qRAM	Setting	Ref.
Hash function	Collision	4/10	2^{120}	2^{16}	-	Classic	[38]
		5/10	2^{120}	2^{64}	-	Classic	[20,37]
		6/10	2^{228}	-	-	Quantum	[25]
		6/10	2^{248}	2^{248}		Classic	[17]
	Preimage	5/10	2^{504}	2^{8}	-		[46]
		6/10	2^{481}	2^{256}	-	Classic	[48]
		7/10	2^{497}	2^{128}			[1]
Compression function	Semi-free-start	5/10	2^{120}	2^{16}	-	Classic	[38]
	Semi-free-start	7/10	2^{184}	2^{8}	-	Classic	[37]
	free-start	8/10	2^{120}	2^{8}	-	Classic	[48]
	free-start	9/10	$2^{220.5}$	-	-	Quantum	Sect. 6
	any	any	2^{256}	-	-	Quantum	[49,25,4]
	any	any	$2^{170.7}$	-	$2^{170.7}$	Quantum	[8]
	any	any	$2^{204.8}$	$2^{102.4}$	-	Quantum	[10]
Saturnin-hash							
Hash	Collision	5/16	2^{64}	2^{66}	-	Classic	Sect. B
		7/16	$2^{113.5}$	-	-	Quantum	Sect. A
	Preimage	7/16	2^{232}	2^{48}	-	Classic	[17]
Compression function	Free-start	6/16	2^{80}	2^{66}	-	Classic	Sect. B
	Semi-free	7/16	$2^{90.99}$	-	-	Quantum	Sect. C
	Free-start	8/16	$2^{122.5}$	-	-	Quantum	Sect. 4
	any	any	2^{128}	-	-	Quantum	[49,25,4]
	any	any	$2^{85.3}$	-	$2^{85.3}$	Quantum	[8]
	any	any	$2^{102.4}$	$2^{51.2}$	-	Quantum	[10]
SKINNY-128-384-MMO/MP							
Compression func.	Free-start	16	$2^{59.8}$	-		Quantum	Sect. 5
	any	any	2^{64}	-		Quantum	[49,25,4]
	any	any	$2^{42.7}$	-	$2^{42.7}$	Quantum	[8]
	any	any	$2^{51.2}$	$2^{25.6}$	-	Quantum	[10]

such that $f(x) = 1$ with $O(\sqrt{2^n/|f^{-1}(1)|})$ calls to the quantum oracle $\mathcal{U}_f$ that outputs $\sum_x a_x |x\rangle |y \oplus f(x)\rangle$ upon input of $\sum_x a_x |x\rangle |y\rangle$. To be more specific, Grover's algorithm iteratively apply the unitary transformation $(2|\psi\rangle\langle\psi| - I)\mathcal{U}_f$ to the uniform superposition $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\mathbb{F}_2^n} |x\rangle$ of all basis vectors produced by applying the Hadamard transformation $H^{\otimes n}$ to $|0\rangle^{\otimes n}$. During this process, the amplitudes of those values x with $f(x) = 1$ are amplified. Then, a final measurement gives a value x of interest with an overwhelming probability [24].

Quantum Random Access Memories (qRAM). A quantum random access memory (qRAM) uses n-qubit to address any quantum superposition of 2^n memory cells. For a list of classical data $L = \{x_0, \cdots, x_{2^n-1}\}$ with $x_i \in \mathbb{F}_2^m$, the qRAM for L is modeled as an unitary transformation $\mathcal{U}^L_{\mathsf{qRAM}}$ such that

$$\mathcal{U}^L_{\mathsf{qRAM}}\left(\sum_i a_i |i\rangle \otimes |y\rangle\right) = \sum_i a_i |i\rangle \otimes |y \oplus x_i\rangle . \tag{1}$$

Currently, it is unknown how a large qRAM can be built. Therefore, quantum algorithms using less or no qRAM are preferred.

2.2 The Full-Active and Non-full-active Super S-box Technique

The super S-box technique proposed by Gilbert et al.[20] and Lamberger et al. [37] extends the Mendel et al.'s [38] inbound part into 2 S-box layers, by identifying four non-interfering $\mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ permutations across two consecutive `AES` rounds and regarding them as four super S-boxes as shown in Figure 3 (a). In [47], Sasaki et al. further reduced the the memory complexity by considering non-full-active super S-boxes as shown in Figure 3 (b).

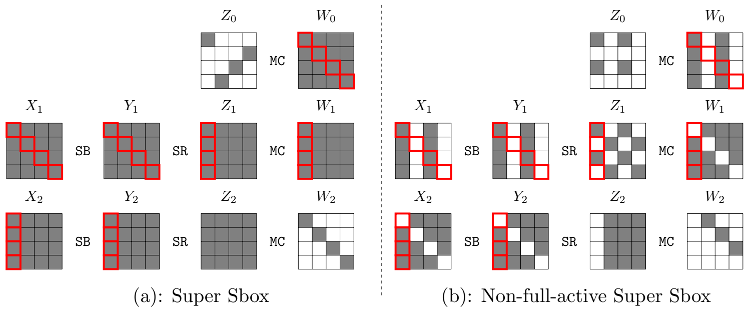


Fig. 3: A differential with non-full-active super S-box

Full-active super S-box. We consider a more general scenario that the internal state of the cipher is a $d \times d$ matrix of c-bit cells. As shown in Figure 3 (a) with $d = 4$, for the ith super S-box $\mathtt{SSB}_i$ and given input difference $\Delta X_1^{(i)}$, we compute $\Delta Y_2^{(i)} = \mathtt{SSB}_i(x \oplus \Delta X_1^{(i)}) \oplus \mathtt{SSB}_i(x)$ for $x \in \mathbb{F}_2^{dc}$. Store the pair $(x, x \oplus \Delta X_1^{(i)})$ in a table $\mathbb{L}^{(i)}[\Delta Y_2^{(i)}]$. In the inbound phase, given $\Delta_{in} = \Delta Z_0$, we compute $\Delta X_1^{(i)}$ for $0 \leq i \leq d-1$, then we compute the d tables $\mathbb{L}^{(0)}$, $\mathbb{L}^{(1)}$, ..., $\mathbb{L}^{(d-1)}$. For each $\Delta_{out} = \Delta W_2 \in \mathbb{F}_2^{dc}$, compute $\Delta Y_2^{(i)}$ with $0 \leq i \leq d-1$ to access the table $\mathbb{L}^{(i)}[\Delta Y_2^{(i)}]$ to generate a pair conforming the truncated differential of the inbound part. Hence, for given Δ_{in}, we need $d \times 2^{dc}$ memory to store the four tables, and will generate $|\Delta_{out}| = 2^{dc}$ pairs on average satisfying the inbound part.

At EUROCRYPT 2020, Hosoyamada and Sasaki [25] converted the classical super S-box technique into a quantum one. They introduced two quantum ways.

The first one is to use the qRAM to replace the classical memory to store the super S-box, which needs a exponential size of qRAM. The second one is to apply the Grover's algorithm to search a conforming pair for a given input-output difference $(\Delta X_1^{(i)}, \Delta Y_2^{(i)})$ of $\texttt{SSB}_i$. This method needs about $2^{dc/2}$ super S-box computations to find the right pair.

Non-full-active super S-box. For the non-full-active super S-box in Figure 3 (b), the Property 1 of MDS in $\texttt{MC}$ is used. Look at $\Delta W_1 = \texttt{MC}(\Delta Z_1)$, suppose there are totally s non-active cells $(s < d)$ and $2d - s$ active cells in ΔZ_1 and ΔW_1 ($s = 3$ in Figure 3 (b)), then by guessing the differences of $d-s$ active cells, we can determine other differences according to Property 1. Then, for a fixed input-output differences $(\Delta X_1^{(i)}, \Delta Y_2^{(i)})$ of $\texttt{SSB}_i$, we can deduce all the input-output differences for the $2d - s$ active cells of two S-box layers for each guess and then deduce their values by accessing the differential distribution table ($\texttt{DDT}$) of the S-box. Now, for the equation $W_1 = \texttt{MC}(Z_1)$, we have $2d - s$ known cells in W_1 and Z_1, hence it acts of probability $2^{-(2d-s-d)c} = 2^{(s-d)c}$. Hence, for a fixed $(\Delta X_1^{(i)}, \Delta Y_2^{(i)})$, we get $2^{(d-s)c} \cdot 2^{(s-d)c} = 1$ conforming pair on average. The time complexity is $2^{(d-s)c}$. The memory is 2^{2c} to store the $\texttt{DDT}$ of S-box.

Property 1. $\texttt{MC} \cdot (Z[1], Z[2], \cdots, Z[d])^T = (W[1], W[2], \cdots W[d])^T$ can be used to fully determine the remaining unknowns if any d cells of Z, W are known.

In the quantum setting, Dong et al. [18] converted the non-full-active super S-box technique into a quantum one by searching the $2^{(d-s)c}$ differences with Grover's algorithm, which gains a square root speedup. Both in quantum and classical setting, the complexity is determined by the number of inactive cells in $(\Delta X_1^{(i)}, \Delta Y_2^{(i)})$, i.e., s.

2.3 Inbound Part with Three Full Rounds

As shown in Figure 4, given fixed differences ΔZ_0 and ΔW_3, Jean et al. [29] introduced an algorithm to find the pairs of conforming to the 3-round differential. At EUROCRYPT 2020, Hosoyamada and Sasaki [25] introduced a memoryless variant as shown in Algorithm 9 given in Supplementary Material E. The time complexity of Algorithm 9 is $2^{d^2c/2+dc}$ and there expect one conforming pair as output. Hosoyamada and Sasaki [25] also introduced the quantum variant, which will be discussed in detail in Section 6.

3 Modeling Rebound Attacks in the Related-key Setting

In the related-key setting, taken MMO mode as an example in Figure 1, we construct free-start collisions using related-key truncated differential trail of E_K, which meets Equation (2):

$$(m \oplus E_K(m)) \oplus (m \oplus \Delta m \oplus E_{K\oplus\Delta K}(m \oplus \Delta m)) = \Delta m \oplus \Delta m = 0. \quad (2)$$

The procedures of the related-key rebound attack are:

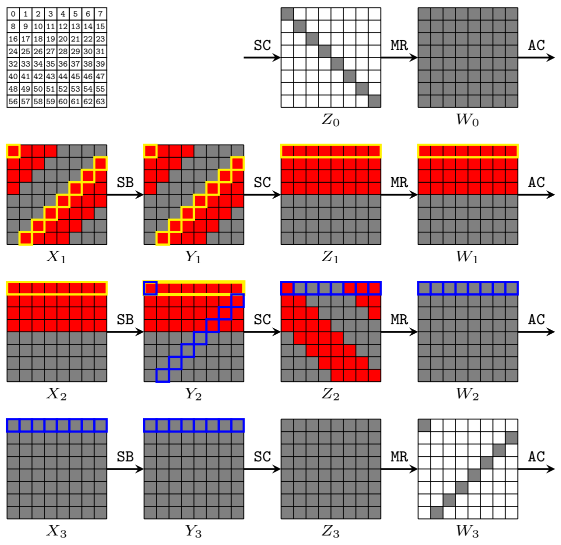


Fig. 4: Details in inbound phase covering 3 rounds

1. Find a related-key truncated differential for E_K,
2. Choose a key pair (K, K') which meets the differential in the key-schedule,
3. Perform the rebound attack in the encryption data path with (K, K').

The Outbound Phase. In the single-key setting, previous works [25,18] consider the probability of the truncated differential, which is mainly due to the cancellations of MC operation. In the related-key setting, we try to use similar method directly, i.e., calculating the probability of differential transition by counting the number of inactive cells in the output of linear operations (e.g. MC, AK etc.) whose input is active. We use the round function of AES as an example without the SR. In Figure 5(a), the four cells in first column of Y_i are active which are the input to the MC operation. The first column of Z_i has one inactive cell. Assume the differences in all active cells are independent uniform random, then $Prob(Y_i \to Z_i) \approx 2^{-c}$ (one cell of the state is of c bits). Similarly, $Prob(Z_i \to W_i) \approx 2^{-c}$. Thus the probability of the truncated differential trail in Figure 5(a) is about 2^{-2c}.

The method borrowed from single-key rebound attack seems to work well, but in related-key setting, this method may lead to a lower probability than the reality. For example, in Figure 5(b), two active cells are cancelled by AK operation. Using the above method, we can calculate the probability of the trail is about 2^{-2c}. Note that in the related-key rebound attack, the key pair is first determined, then perform the rebound attack in the encryption data path, where key materials act as constants. Hence, the probability of the outbound phase in the encryption data path is computed under a fixed key difference. Therefore, $\Delta K_i[0, 1] = \Delta Z_i[0, 1]$ and $\Delta Z_i[0, 1]$ is fixed. Due to Property 1, all other

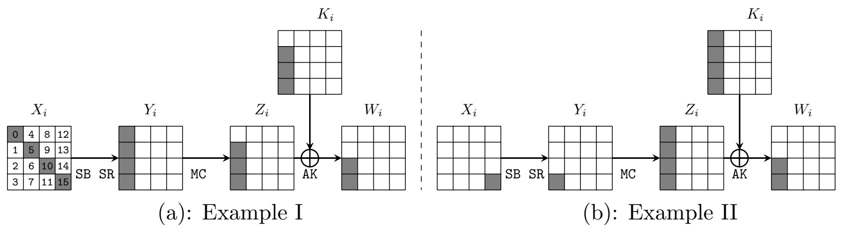


Fig. 5: AES rounds in forward outbound phase.

active cells of differences in Y_i and Z_i are determined. Hence, the probability of the differential is determined by the differential propagation of the S-box, i.e., $Prob(\Delta X_i[15] \xrightarrow{S\text{-}box} \Delta Y_i[3]) > 2^{-c}$ with DDT, which is bigger than 2^{-2c}. In detail, we derive the relationship between the first column of Y_i and Z_i from MC as shown in Equation (3).

$$\begin{bmatrix} \Delta Z_i[0] \\ \Delta Z_i[1] \\ \Delta Z_i[2] \\ \Delta Z_i[3] \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \times \begin{bmatrix} \Delta Y_i[0] \\ \Delta Y_i[1] \\ \Delta Y_i[2] \\ \Delta Y_i[3] \end{bmatrix}. \tag{3}$$

As three cells in the first column of ΔY_i are 0 and $\Delta Z_i[0,1] = \Delta K_i[0,1]$, we have

$$\begin{bmatrix} \Delta K_i[0] \\ \Delta K_i[1] \\ \Delta Z_i[2] \\ \Delta Z_i[3] \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Delta Y_i[3] \end{bmatrix}, \tag{4}$$

which shows that $\Delta Y_i[3] = \Delta K_i[0] = \Delta K_i[1]$. Hence, $\Delta K_i[0]$ and $\Delta K_i[1]$ are related to each other. We call the number of cells in key, whose differences can be chosen independent randomly, the *degree of freedom* in the key differential states. In Figure 5(b), key states have four active cells, among them two cells meet the condition $\Delta K_i[0] = \Delta K_i[1]$ that consumes one-cell degree of freedom. Hence, the degrees of freedom in the key differential states are 4-1=3 in Figure 5(b). Therefore, the degree of freedom in K is reduced to increase the probability of the trail in Figure 5(b) from 2^{-2c} to about 2^{-c}. The consumption of freedom in the whole differential trail should not be higher than the number of active cells in key. Note that similar technique has already been used by Cid et al. [12] in the cryptanalysis of Deoxys against related-key differential attack. We apply the technique to the rebound attack by taking the features of rebound attack into the model.

Degree of Freedom. For a target with linear key schedule algorithms (e.g. Saturnin [9] and SKINNY [3]), we formulate its degree of freedom in the following. Taking Saturnin as an example, if there are t active cells in the master key, then we say that the initial degree of freedom for the key difference is t-cell (denoted by DoK=t), since there are about $(2^c)^t$ different choices for the key difference. However, as discussed previously, in rebound attacks exploiting related-key differentials, we may constrain the key difference by a system of linear equations with the active cells in the master key as variables to increase the probability of the outbound differentials. Assuming we have l independent linear equations, then l-cell degree of freedom is consumed (denoted by $\mathrm{DoK}^- = l$). Therefore, to ensure there is at least one solution for the master key difference, we require $\mathrm{DoK} \geq \mathrm{DoK}^-$. Otherwise, we have an over-defined system of equations for the active cells of the master key, which may have some conflicts.

Besides the degree of freedom from the master key difference, another source of degree of freedom should be considered. For a given master key difference, we can form $(2^c)^{\bar{n}}$ key pairs satisfying the given difference, where the key is of $\bar{n}$ c-bit cells. Taking the encryption data path into account and supposing that for a given $(\Delta_{in}, \Delta_{out})$ and key pair (K, K'), there is one solution for the inbound part in the data encryption path on average, then we can generate $(2^c)^{\mathrm{DoK}-\mathrm{DoK}^-+\bar{n}}(|\Delta_{in}| \cdot |\Delta_{out}|)$ starting points as $(K, M, K'M')$, which is called the degrees of freedom for the rebound attack [38] (denoted by DoA). To expect one solution fulfilling the outbound differential with probability p, we require that

$$(2^c)^{\mathrm{DoA}} = (2^c)^{\mathrm{DoK}-\mathrm{DoK}^-+\bar{n}}(|\Delta_{in}| \cdot |\Delta_{out}|) \geq \frac{1}{p}. \tag{5}$$

3.1 Dedicated Modelings and Case Study on `Saturnin-hash`

`Saturnin` is a suite of lightweight symmetric algorithms proposed by Canteaut et al. [9]. It is among the 2nd round candidates of the NIST LWC. Based on a 256-bit `AES`-like block cipher with 256-bit key, two authenticated ciphers and a hash function are designed. In this section, we focus on its hash function, called `Saturnin-Hash`. The round function only consists of `AK`, `SB` layer and linear layer, where `MixRows (MR)` and `MixColumns (MC)` are applied alternatively in even or odd number of round. The key schedule is linear and simple. In even round, K is used and in odd round the K is rotated by 5 cells (denoted as $\tilde{K}$).

Related-key Truncated Differential Model. For an R-round primitive, we use several binary variables $x_r^{i,j}$ and $y_r^{i,j}$ to represent the state before and after the `MR` (or `MC`) operations in the r-th round, where i and j mean that the cell is in i-th row and j-th column. These variables are 1 if and only if the corresponding cell is active. For the key states, we use $K^{i,j}$ and $\tilde{K}^{i,j}$ to represent the rotated key and the master key in the same way.

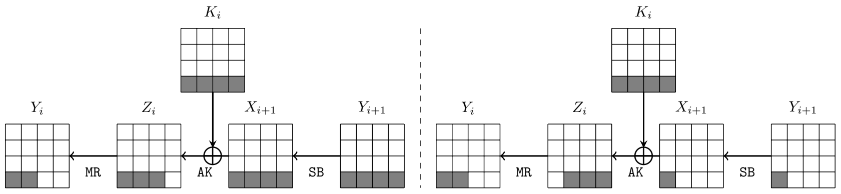
Without loss of generality, we only consider `MR` operation now. To model the `MR` operations (similar constraints are also applied to `MC`), we use binary variables b_r^i to express `MR` operations are active or not in the i-th row of r-th round, and use branch number to generate constraints just like Mouha et al.'s model [41].

Another operation is key addition. The constraint of key addition are quite like constraint of XOR, except the result of two active cells addition can be active or inactive.

The Outbound Phase. As shown in Figure 5(b), the number of cancelled cells could not show the real probability in a related-key model. Hence, the constraints in our model are different from single-key models. We use $Prob_r^i$ to represent the probability of the i-th row in round r.

In forward part of the outbound phase, we use c_r^i to represent the number of cells cancelled after the r-th round `MR` operation in row i, and $\tilde{c}_r^i$ to represent the number of cells cancelled after the next key addition operation in row i. If $\sum_{j=0}^{j\leq 3} x_r^{i,j} \geq c_r^i + \tilde{c}_r^i$ (like the trail in Figure 5(a)), then the probability of this `MR` operation in this row is estimated by $c_r^i + \tilde{c}_r^i$ (to show the connection of probabilities and variables in our MILP model, the probabilities are taken in $-log_{2^c}$). If $\sum_{j=0}^{j\leq 3} x_r^{i,j} < c_r^i + \tilde{c}_r^i$ (like the trail in Figure 5(b)), then the probability is $\sum_{j=0}^{j\leq 3} x_r^{i,j}$, and the degree of freedom in key states is consumed $c_r^i + \tilde{c}_r^i - \sum_{j=0}^{j\leq 3} x_r^{i,j}$. Thus,

$$Prob_r^i = \min(c_r^i + \tilde{c}_r^i, \sum_{j=0}^{j\leq 3} x_r^{i,j}). \tag{6}$$



(a): Example III (b): Example IV

Fig. 6: `Saturnin` rounds in backward outbound phase.

Similar to forward part, in backward part, we also use c_r^i to represent the number of cells that are cancelled by the r-th round $\texttt{MR}^{-1}$ operation in row i, and $\tilde{c}_r^i$ to represent the number of cells are cancelled before the next key addition operation in row i. If $\sum_{j=0}^{j\leq 3} x_{r+1}^{i,j} \geq c_r^i$ (like the trail in Figure 6(a)), then the probability of this `MR` operation in this row is $c_r^i + \tilde{c}_r^i$. If $\sum_{j=0}^{j\leq 3} x_{r+1}^{i,j} < c_r^i$(like the trail in Figure 6(b)), then the cancellation of this `MR` operation in this row is $\sum_{j=0}^{j\leq 3} x_{r+1}^{i,j}$, and the degree of freedom in key states is consumed $c_r^i - \sum_{j=0}^{j\leq 3} x_r^{i,j}$. Thus,

$$Prob_r^i = \min(c_r^i + \tilde{c}_r^i, \sum_{j=0}^{j\leq 3} x_{r+1}^{i,j}). \tag{7}$$

To limit the consumption of freedom, we add the following constraint

$$\sum_{Forward} (c_r^i + \tilde{c}_r^i - Prob_r^i) + \sum_{Backward} (c_r^i - Prob_r^i) \leq \sum_{0\leq i,j\leq 3} K^{i,j}. \tag{8}$$

The Inbound Phase. We use a variable l to determine the inbound part and outbound part, and the inbound part includes r_{in} rounds. Thus round $l+1$ to $l+r_{in}$ are inbound part, while other rounds are outbound parts. If $r_{in}=2$, we use the super S-box techniques to solve the inbound part. In classical setting, it usually does not increase the overall time complexity, and only need some memories as shown in Section 2.2. However, in quantum setting without qRAM, the overall time complexity is also affected by the super S-box technique. As shown by Dong et al. [18], if the super S-boxes are not fully active, the time for quantum attack may be reduced. Following the notations in Section 2.2, the number of inactive S-boxes in the i-th super S-box $\texttt{SSB}_i$ is denoted as s^i. Then the quantum time to solve the inbound part is about $\sqrt{2^{d-\min\{s^0,s^1,s^2,s^3\}}}$ according to Dong et al. [18], where d is the number of cells in each row, and $d=4$ for `Saturnin`. In related-key setting, some cells in super S-boxes can be determined by key difference. As we shown in Algorithm 2 of Section 4, cells with known difference play the same role as inactive cells in non-full active super S-boxes technique. Thus s_i denote the number of cells whose difference is fixed before or after the `MR` or `MC` operation in the middle of a super S-box.

When $r_{in}=3$, the inbound phase in solved by the methods of Jean et al. [29] classically or Hosoyamada et al. [25] quantumly. Both the time complexities are fixed and independent to the rebound attack trails as shown in Section 2.3. We will give more details in the attack on `Whirlpool`, whose rebound trail includes a 3-round inbound part in both the key schedule and encryption data path.

Time Complexity and Objective Function. In quantum setting without qRAM, we have two time complexities according to r_{in}:

- $r_{in}=2$, the time complexity is about
$$\sqrt{2^{(\sum Prob_r^i+\sum x_0^{i,j}+d-\min\{s^0,s^1,s^2,s^3\})}}, \tag{9}$$
where $\sum Prob_r^i$ corresponds to the probability of the truncated difference of the outbound phase, $\sum x_0^{i,j}$ are the number of active cells to be collided for the plaintext and ciphertext, $d-\min\{s^0,s^1,s^2,s^3\}$ corresponds to the time to solve the inbound part. Hence, when $r_{in}=2$, the objective function is to minimize
$$\sum Prob_r^i+\sum x_0^{i,j}+d-\min\{s^0,s^1,s^2,s^3\}. \tag{10}$$
- $r_{in}=3$, the objective function is
$$\sum Prob_r^i+\sum x_0^{i,j}. \tag{11}$$

The Incompatibilities within Many Rounds. Cid et al. [12] described an MILP model to search the related-key differentials on ciphers with linear key schedule, e.g., `Deoxys-BC` [32]. Since the relationship between `Deoxys-BC`'s round keys are somewhat weakened by the LFSRs, they do not need to consider incompatibilities between many rounds. In `Saturnin`, the round keys are identical for many rounds, which lead to strong relationship on the round keys.

Though we limit the consumption of degree of freedom in our MILP model, a trail can be incompatible when the same key cell needs to satisfy two different relationships in different rounds. For example, in Figure 7, from Y_2 to X_3 we have $(\Delta Z_2[2], \Delta k_{11}, \Delta k_{15}, \Delta k_0) = \texttt{MR}(\Delta Y_2[2], 0, 0, 0)$. From Y_4 to X_5 we have $(\Delta k_7, \Delta k_{11}, \Delta k_{15}, \Delta k_0) = \texttt{MR}(\Delta Y_4[2], 0, 0, 0)$. The above two linear equations have 6 same cells., Due to Property 1, $\Delta Z_2[2] = \Delta k_7$, then $\Delta X_3[2]$ should be 0, which is a contradiction.

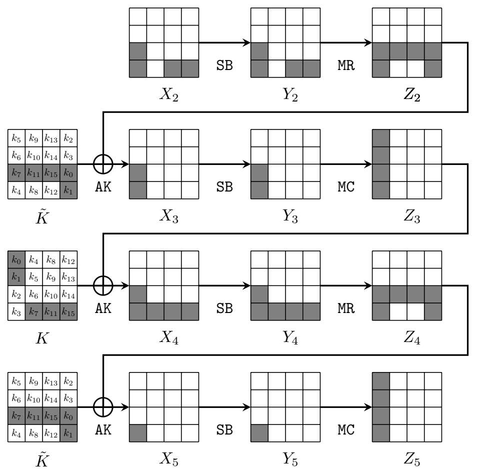


Fig. 7: An incompatible trail of `Saturnin`

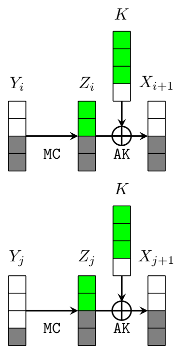


Fig. 8: An incompatible trail

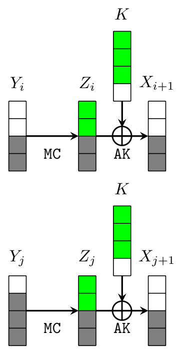


Fig. 9: A compatible trail

Adding more constraints to remove this kind of contradictions in MILP model is quite hard. According to Property 1, a set of equations of `MC` (or `MR`) operation has 8 cells of variables, and if two sets of equations have at least 4 same cells, then all cells of variables in the two sets of equations should be same. We use this property to fast delete the incompatible trails. Figure 8 and Figure 9 show examples of incompatible and compatible trails. The inactive cells are in white and active cells are in gray and green, and the difference in green cells are determined by key difference. We can encode the truncated difference in Y and Z to a 8-dimensional vector $\mathbb{S} = (y_0, y_1, y_2, y_3, z_0, z_1, z_2, z_3)$, where $y_m = 1$ if $\Delta Y[m]$ is inactive with $0 \leq m \leq 3$, else $y_m = 0$; $z_m = 1$ if $\Delta Z[m]$ is 0 or equals to key differences, else $z_m = 0$. For example, in Figure 8, we have $\mathbb{S}_i = (1,1,0,0,1,1,0,0)$ for round i and $\mathbb{S}_i = (1,1,1,0,1,1,0,0)$ for round j with the same K. The dot product of two vectors $\mathbb{S}_i$ and $\mathbb{S}_j$ is the number of same cells of two sets of equations of `MC` (or `MR`) operations. For example, $\langle \mathbb{S}_i, \mathbb{S}_j \rangle = 4$ in Figure 8, hence, due to Property 1, all the cells of differences in Y_i and Y_j (also for Z_i, Z_j and X_{i+1}, X_{j+1}) should be the same. However, $Y_i[2]$ is active but $Y_j[2]$ is inactive, which leads to contradiction and Figure 8 is an incompatible trail. In Figure 9, we have $\mathbb{S}_i = (1,1,0,0,1,1,0,0)$ and $\mathbb{S}_j = (1,0,0,0,1,1,0,0)$ with $\langle \mathbb{S}_i, \mathbb{S}_j \rangle = 3 < 4$, hence the trail is compatible.

Since we can derive the vector $\mathbb{S}_i$ from the solutions of our MILP model, we use the PoolSearchMode of Gurobi to get many solutions for our MILP model and then check if one of the solutions does not have this kind of contradiction. For 8-round `Saturnin` with $l \geq 1$ and $r_{in} = 2$, we get thousands of different truncated differentials from our MILP model through the PoolSearchMode and after checking them with the above method, none of them are left; for 8-round `Saturnin` with $l = 0$ and $r_{in} = 2$, we get a hundred of different truncated differentials and most of them are compatible. For those left solutions, we pick one trail to launch our rebound attacks. See supplementary materials for the source code of constructing MILP model and detecting contradiction. We have put the source code for the automatic model of `Saturnin-hash` in a public domain:

https://github.com/rebound-rk/rebound-rk

4 Free-Start Collision on 8-round `Saturnin-hash`

By applying the MILP model, we find an 8-round truncated differential on `Saturnin` as shown in Figure 10(a). We perform the quantum collision attack based on the truncated differential. The inbound phase covers from Y_0 to X_3, including two `SB` layers. The two outbound phases are from Y_0 to the plaintext and X_3 to the ciphertext.

In the inbound phase, there are four parallel non-full active super S-boxes. The input difference $\Delta_{in} = \Delta X_1$ is determined by ΔY_0. At round 2 and 3, from $\texttt{MR}(\Delta Y_2) \oplus \Delta\tilde{K} = \Delta X_3$, at the 3rd row, we get

$$\texttt{MR}^{-1}(\Delta k_7, \Delta k_{11}, \Delta k_{15}, \Delta k_0) = (\Delta Y_2[2], 0, 0, 0). \tag{12}$$

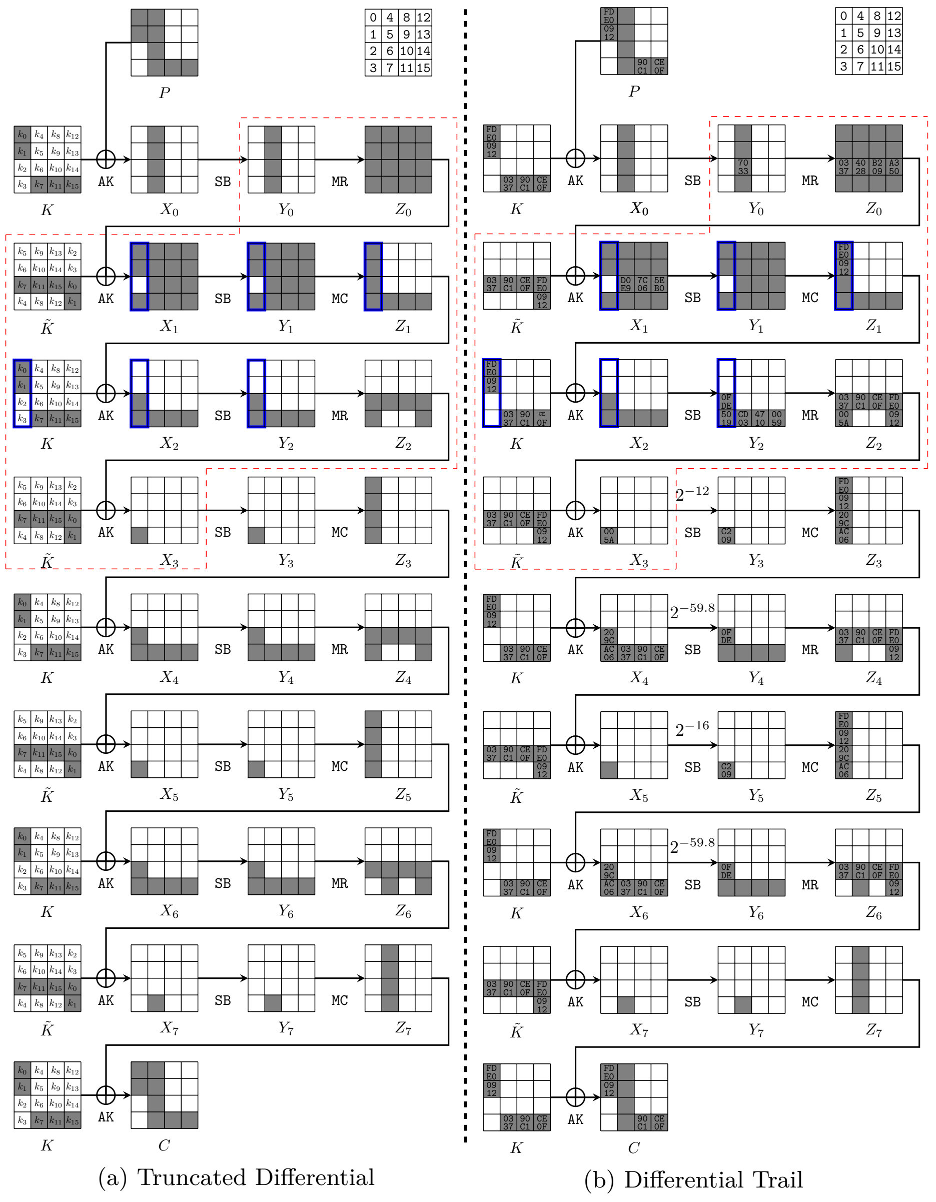


Fig. 10: 8-round Related-key Rebound-attack Trail on `Saturnin-hash`

For row 3 of the computation from ΔY_4 to ΔX_5, and from ΔY_6 to ΔX_7, the same requirement of Equation (12) is also applied, since the subkeys are all $\tilde{K}$ and the truncated form are the same.

At round 3 to 4, in the first column of the computation from ΔY_3 to ΔX_4, we have $\Delta Z_3[0] = \Delta k_0$ and $\Delta Z_3[1] = \Delta k_1$. Further, we get

$$\mathtt{MC}^{-1}\begin{pmatrix}\Delta k_0\\ \Delta k_1\\ \Delta Z_3[2]\\ \Delta Z_3[3]\end{pmatrix} = \begin{pmatrix}0\\0\\0\\ \Delta Y_3[3]\end{pmatrix}. \tag{13}$$

The condition of Equation (13) also applies to the computation from ΔY_5 to ΔX_6.

Hence, for Equation (13) and (12), if we fixed Δk_0, then Δk_7, Δk_{11}, Δk_{15}, and Δk_1 are determined by Property 1.

As shown in Figure 10(a), at round 0 and 1, $\Delta X_1[2] = \Delta k_7 \oplus \Delta Z_0[2] = 0$, so $\Delta Z_0[2] = \Delta k_7$. From $\Delta Z_0 = \mathtt{MR}(\Delta Y_0)$, for the third row, we have $\Delta Y_0[2, 10, 14] = 0$ and $\Delta Z_0[2] = \Delta k_7$ and Equation (14) is derived. Hence, if Δk_7 is fixed, all other differences in the active cells of ΔY_0 and ΔZ_0 in row 3 are deduce by Property 1.

$$\mathtt{MR}^{-1}(\Delta k_7, \Delta Z_0[6], \Delta Z_0[10], \Delta Z_0[14]) = (0, \Delta Y_0[6], 0, 0). \tag{14}$$

Algorithm 1: Determine the Differences From the Truncated Form

```
1  for $\Delta k_0 \in \mathbb{F}_2^{16}$ do
2      Deduce $\Delta k_7$, $\Delta k_{11}$, $\Delta k_{15}$, $\Delta k_1$ by Equation (13) and (12) and Property 1
3      /* All the differences in the key schedule are determined.    */
4      Round 2: Deduce $\Delta Z_2[2, 6, 10, 14]$, $\Delta Y_2[2]$
5      Round 3: Deduce $\Delta Y_3[3]$ by Equation (13) and Property 1. Then
         $Z_3[0, 1, 2, 3]$ and $X_4[2, 3]$ are fixed.
6      Round 4: Similar to Round 2 to get $\Delta Z_4[2, 6, 10, 14]$, $\Delta Y_4[2]$. In addition,
         we have $\Delta Z_4[15] = \Delta k_1$
7      Round 5: Similar to Round 3 to get $\Delta Y_5[3]$, $Z_5[0, 1, 2, 3]$ and $X_6[2, 3]$
8      Round 6: Similar to Round 2 to get $\Delta Z_6[2, 6, 10, 14]$, $\Delta Y_6[2]$. In addition,
         we have $\Delta Z_6[15] = \Delta k_1$
9      Round 0: With Equation (14), we deduce $\Delta Z_0[6, 10, 14]$ and $\Delta Y_0[6]$. Then
         $\Delta X_1[6, 10, 14]$ are determined.
10     Round 1: Since $\Delta Z_1[0] = \Delta k_0$ and $\Delta Z_1[1] = \Delta k_1$, $\Delta Z_1[0]$ and $\Delta Z_1[1]$ is
         fixed.
11     Round 7: In Saturnin-hash (MMO hashing mode), the plaintext is
         XORed into the ciphertext of the internal block cipher to output the
         digest. We have $T = P \oplus C = X_0 \oplus K \oplus Z_7 \oplus K = X_0 \oplus Z_7$. Then, if two
         message collide, we have $\Delta T = 0 = \Delta X_0 \oplus \Delta Z_7$.
12     As shown in Figure 10(b), from $\Delta X_4$ to $\Delta Y_5$, multiple differential trails
         are taken into account.
```

We derive an 8-round rebound-attack characteristic in Figure 10(b) from the truncated form in Figure 10(a) by Algorithm 1. By traversing $\Delta k_0 \in \mathbb{F}_2^{16}$ in

Algorithm 1, we find characteristic with as higher probability as possible. The best trail is given in Figure 10(b), whose total probability of the outbound phase is $2^{-(12+59.8+16+59.8+64)} = 2^{-211.6}$ including the probability of $\Delta X_0 = \Delta Z_7$. In round 4, $2^{-59.8}$ is the total probability of a cluster differential trails from ΔX_4 to ΔZ_4. The same happens to round 6.

As shown in Figure 10(b), the 2nd, 3rd and 4th super S-boxes are typical non-full-active super S-boxes, where there are only 5 active cells among the 8 input-ouput cells between `MC` in round 1 in each super S-box. However, the first super S-box is not a typical one. In fact, between `MC` in round 1 in the first Super-Sbox, there are one non-active cell and two cells with fixed differences. However, we can regard the two cells with fixed differences as another two "non-active" cells to perform the quantum version of non-full active super S-box technique [18] whose details will be given in Algorithm 2.

For the ith ($i = 0, 1, 2, 3$) non-full-active super S-box, we define $G^{(i)} : \mathbb{F}_2^{16} \times \mathbb{F}_2^3 \mapsto \mathbb{F}_2$ as $G^{(i)}(K, K', \Delta X_1^{(i)}, \Delta Y_2^{(i)}; x, \beta)$, where $x = X_1^{(i)}[0] \in \mathbb{F}_2^{16}$ and $\beta = \beta_0 \| \beta_1 \| \beta_2 \in \mathbb{F}_2^3$. $G^{(i)}(K, K', \Delta X_1^{(i)}, \Delta Y_2^{(i)}; x, \beta) = 1$ if and only if (x, β) leads to a valid connection of $(\Delta X_1^{(i)}, \Delta Y_2^{(i)})$ under the key pair (K, K'). The quantum implementation of $\mathcal{U}_{G^{(0)}}$ is given in Algorithm 2.

Complexity of $\mathcal{U}_{G^{(0)}}$ is given in Algorithm 2. The time is bounded by Line 7 to Line 9 of Algorithm 2, which is about (including uncomputing)

$$3 \times \frac{\pi}{4} \cdot \sqrt{2^{16}} \cdot 2 \cdot 2 = 2^{11.24} \quad \text{Sbox evaluations.} \tag{15}$$

Since the probability of the outbound phase is $2^{-211.6}$, after traversing $2^{211.6}$ starting points computed by the inbound phase, it is expected to find one collision. Given the key difference $\Delta K = K \oplus K'$, there are 2^{256} valid key pairs (K, K'). Hence, we have enough degrees of freedom to find the collision. For simplicity, we just fix the input difference ΔX_1 of the inbound phase and compute the starting points by traversing a 212-bit K to find the collision. Define $F : \mathbb{F}_2^{212} \times \mathbb{F}_2^3 \mapsto \mathbb{F}_2$ as $F(\Delta K, \Delta X_1, \Delta Y_2; x, \alpha)$, where $x = K \in \mathbb{F}_2^{212}$ and $\alpha = \alpha_0 \| \alpha_1 \| \alpha_2 \in \mathbb{F}_2^3$. $F(\Delta K, \Delta X_1, \Delta Y_2; x, \alpha) = 1$ if and only if $(\Delta K, \Delta X_1, \Delta Y_2; x, \alpha)$ leads a collision. The implementation of $\mathcal{U}_F$ is given in Algorithm 3.

Complexity of $\mathcal{U}_F$ in Algorithm 3. There are four Grover searches on $G^{(i)}$ in Line 4 and 8. There are four calls of Algorithm 2 in Line 5 and Line 9. Those procedures bound the time complexity of $\mathcal{U}_F$ as

$$4 \cdot \frac{\pi}{4} \cdot \sqrt{2^{19}} \cdot 2^{11.24} + 4 \cdot 2^{11.24} = 2^{22.39} \quad \text{S-box evaluations.} \tag{16}$$

To find the collision on 8-round `Saturnin-hash`, we apply Grover search on $F(\Delta K, \Delta X_1, \Delta Y_2; \cdot) : \mathbb{F}_2^{212+3} \mapsto \mathbb{F}_2$ with $\mathcal{U}_F$ in Algorithm 3, which costs

$$\frac{\pi}{4} \cdot \sqrt{2^{212+3}} \cdot 2^{22.39} = 2^{129.54} \quad \text{S-box evaluations.} \tag{17}$$

Algorithm 2: Implementation of $\mathcal{U}_{G^{(0)}}$ without using qRAMs

```
Input: $|K, K', \Delta X_1^{(0)}, \Delta Y_2^{(0)}; X_1^{(0)}[0], \beta\rangle\, |y\rangle$ with $\beta = (\beta_0, \beta_1, \beta_2) \in \mathbb{F}_2^3$
Output: $|K, K', \Delta X_1^{(0)}, \Delta Y_2^{(0)}; X_1^{(0)}[0], \beta\rangle\, |y \oplus G^{(0)}(K, K', \Delta X_1^{(0)}, \Delta Y_2^{(0)}; X_1^{(0)}[0], \beta)\rangle$
1  /* Please focus on the super Sbox marked by blue box in Figure 10
      */
2  $Y_1^{(0)}[0] \leftarrow S(X_1^{(0)}[0])$
3  $\Delta Y_1^{(0)}[0] \leftarrow S(X_1^{(0)}[0] \oplus \Delta X_1^{(0)}[0]) \oplus S(X_1^{(0)}[0])$
4  Solving the system of equations $\mathtt{MC}(\Delta Y_1^{(0)}) = \Delta Z_1^{(0)}$ with the knowledge of
     $\Delta Z_1^{(0)}[0] = \mathtt{0xFDE0}$, $\Delta Z_1^{(0)}[1] = \mathtt{0x0912}$ and $\Delta Y_1^{(0)}[2] = 0$
5  /* All differences of cells in $\Delta Y_1^{(0)}$, $\Delta Z_1^{(0)}$ are known */
6  Let $g_j : \mathbb{F}_2^{16} \rightarrow \mathbb{F}_2$ be a Boolean function such that $g_j(\delta_{in}, \delta_{out}, \beta_j = 0; x) = 1$ if
     and only if $S(x) \oplus S(x \oplus \delta_{in}) = \delta_{out}$ and $x \leq x \oplus \delta_{in}$, and
     $g_j(\delta_{in}, \delta_{out}, \beta_j = 1, x) = 1$ if and only if $S(x) \oplus S(x \oplus \delta_{in}) = \delta_{out}$, and
     $x > x \oplus \delta_{in}$.
7  Run the Grover search on the function $g_0(\Delta X_1^{(0)}[1], \Delta Y_1^{(0)}[1], \beta_0; \cdot) : \mathbb{F}_2^{16} \rightarrow \mathbb{F}_2$.
     Let $X_1^{(0)}[1]$ be the output.
8  Run the Grover search on the function $g_1(\Delta X_1^{(0)}[3], \Delta Y_1^{(0)}[3], \beta_1; \cdot) : \mathbb{F}_2^{16} \rightarrow \mathbb{F}_2$.
     Let $X_1^{(0)}[3]$ be the output.
9  Run the Grover search on the function $g_2(\Delta X_2^{(0)}[3], \Delta Y_2^{(0)}[3], \beta_2; \cdot) : \mathbb{F}_2^{16} \rightarrow \mathbb{F}_2$.
     Let $X_2^{(0)}[3]$ be the output.
10 Compute $Y_1^{(0)}[1]$, $Y_1^{(0)}[3]$ and $Z_1^{(0)}[3]$ ;     /* $Y_1^{(0)}[0]$ is known */
11 Solve the equation $\mathtt{MC}(Y_1^{(0)}) = Z_1^{(0)}$ for other unknown cells, i.e., $Y_1^{(0)}[2]$,
     $Z_1^{(0)}[0, 1, 2]$, and $X_1^{(0)}$
12 /* the value $Y_1^{(0)}$ is known */
13 if $S(Z_1^{(0)}[2] \oplus \Delta Z_1^{(0)}[2] \oplus K'[2]) \oplus S(Z_1^{(0)}[2] \oplus K[2]) = \Delta Y_2^{(0)}[2]$ then
14     return $|K, K', \Delta X_1^{(0)}, \Delta Y_2^{(0)}; X_1^{(0)}[0], \beta\rangle\, |y \oplus 1\rangle$
15 else
16     return $|K, K', \Delta X_1^{(0)}, \Delta Y_2^{(0)}; X_1^{(0)}[0], \beta\rangle\, |y\rangle$
```

Since there are $16 \times 8 = 128$ Sbox applications, the time complexity to find the collision is about $2^{129.54}/128 = 2^{122.54}$ 8-round `Saturnin-hash`.

Besides the 8-round free-start collision attack, we also identify a 7-round single-key rebound attack trail for `Saturnin-hash`, which leads a 7-round collision attack on `Saturnin-hash` in Supplementary Material A.

Algorithm 3: Implementation of $\mathcal{U}_F$ without using qRAMs

```
Input: $|\Delta K, \Delta X_1, \Delta Y_2; K, \alpha\rangle\, |y\rangle$ with $\alpha = (\alpha_0, \alpha_1, \alpha_2) \in \mathbb{F}_2^3$
Output: $|\Delta K, \Delta X_1, \Delta Y_2; K, \alpha\rangle\, |y \oplus F(\Delta K, \Delta X_1, \Delta Y_2; K, \alpha)\rangle$
1  Compute $K' = K \oplus \Delta K$
2  for $i \in \{0, 1, 2\}$ do
3      Derive the $\Delta X_1^{(i)}$ and $\Delta Y_2^{(i)}$ for SSB$^{(i)}$ from the $\Delta X_1$ and $\Delta Y_2$
4      Run Grover search on the function $G^{(i)}(K, K', \Delta X_1^{(i)}, \Delta Y_2^{(i)}; \cdot) : \mathbb{F}_2^{19} \mapsto \mathbb{F}_2$.
         Let $X_1^{(i)}[0] \in \mathbb{F}_2^{16},\ \beta^{(i)} \in \mathbb{F}_2^3$ be the output.
5      Run Line 2 to Line 11 of Algorithm 2 with $X_1^{(i)}[0] \in \mathbb{F}_2^{16},\ \beta^{(i)} \in \mathbb{F}_2^3$ as
         input. Let $X_1^{(i)}$ as ouput.
6      Let $\tilde{X}_1^{(i)} = \max\{X_1^{(i)}, X_1^{(i)} \oplus \Delta X_1^{(i)}\}$ if $\alpha_i = 0$, else
         $\tilde{X}_1^{(i)} = \min\{X_1^{(i)}, X_1^{(i)} \oplus \Delta X_1^{(i)}\}$
7  Derive the $\Delta X_1^{(3)}$ and $\Delta Y_2^{(3)}$ for SSB$^{(3)}$ from the $\Delta X_1$ and $\Delta Y_2$
8  Run Grover search on the function $G^{(3)}(K, K', \Delta X_1^{(3)}, \Delta Y_2^{(3)}; \cdot) : \mathbb{F}_2^{19} \mapsto \mathbb{F}_2$.
     Let $X_1^{(3)}[0] \in \mathbb{F}_2^{16},\ \beta^{(3)} \in \mathbb{F}_2^3$ be the output.
9  Run Line 2 to Line 11 of Algorithm 2 with $X_1^{(3)}[0] \in \mathbb{F}_2^{16},\ \beta^{(3)} \in \mathbb{F}_2^3$ as input.
     Let $X_1^{(3)}$ as ouput.
10 Let $\tilde{X}_1^{(3)} = \max\{X_1^{(i)}, X_1^{(i)} \oplus \Delta X_1^{(i)}\}$
11 /* Create the starting point $(K, X_1)$ with $(\Delta K, \Delta X_1, \Delta Y_2)$ */
12 $X_1 \leftarrow (\tilde{X}_1^{(0)}, \tilde{X}_1^{(1)}, \tilde{X}_1^{(2)}, \tilde{X}_1^{(3)})$
13 $X_1' \leftarrow X_1 \oplus \Delta X_1$
14 Compute forward and backward to the beginning and ending of the 8-round
     trail from $(X_1, X_1')$ with $(K, K')$
15 if $(X_1, X_1')$ and $(K, K')$ lead to a collision then
16     return $|\Delta K, \Delta X_1, \Delta Y_2; K, \alpha\rangle\, |y \oplus 1\rangle$
17 else
18     return $|\Delta K, \Delta X_1, \Delta Y_2; K, \alpha\rangle\, |y\rangle$
```

5 Free-Start Collision on Hashing Modes with SKINNY-n-$3n$

SKINNY is a family of lightweight block ciphers designed by Beierle et al. [3]. In this section, we apply our method to SKINNY-n-$3n$. The overall structure of SKINNY-n-$3n$ and its round function are given in Figure 11. The MC operation is non-MDS, which meets:

$$\texttt{MC}\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \oplus c \oplus d \\ a \\ b \oplus c \\ a \oplus c \end{pmatrix} \text{ and } \texttt{MC}^{-1}\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \beta \\ \beta \oplus \gamma \oplus \delta \\ \beta \oplus \delta \\ \alpha \oplus \delta \end{pmatrix}. \quad (18)$$

Since SKINNY applies non-MDS matrix in MC, we will adapt the method of super S-box technique for SKINNY. Different from the super S-box technique with MDS matrix [20,37], we do not need to an exponential memory to store the

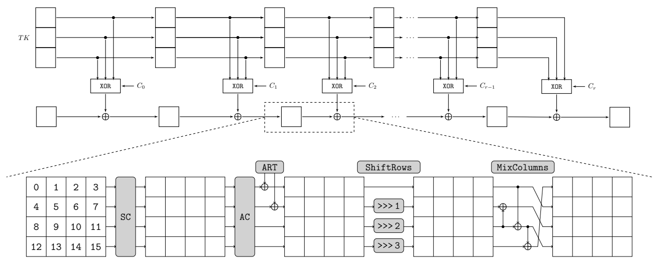


Fig. 11: The high-level structure of SKINNY-n-$3n$ and its round function (Thanks to https://www.iacr.org/authors/tikz/)

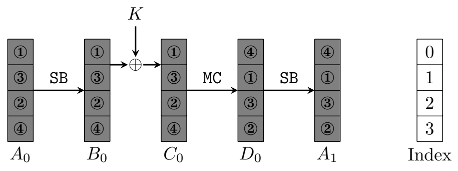


Fig. 12: Super S-box with SKINNY's non-MDS matrix

differential distribution of the super S-box, which is very friendly to quantum attacks.

5.1 Super S-box with Non-MDS Matrix

As shown in Figure 12 (SR is omitted), the circled numbers indicate the computation sequence. When computing the super S-box, the key pair is fixed, i.e., K and K' are known.

1. In step ①, we have $D_0[1] = C_0[0]$ due to Equation (18), then we have

$$\begin{aligned}
&\Delta A_1[1] = \mathtt{S}(D_0[1]) \oplus \mathtt{S}(D_0'[1])\\
&= \mathtt{S}(C_0[0]) \oplus \mathtt{S}(C_0'[0])\\
&= \mathtt{S}(\mathtt{S}(A_0[0]) \oplus K[0]) \oplus \mathtt{S}(\mathtt{S}(A_0'[6]) \oplus K'[0])\\
&= \mathtt{S}(\mathtt{S}(A_0[0]) \oplus K[0]) \oplus \mathtt{S}(\mathtt{S}(A_0[0] \oplus \Delta A_0[0]) \oplus K'[0]).
\end{aligned} \tag{19}$$

   Hence, given input-output differeces $(\Delta A_0[0], \Delta A_1[1])$, we compute one conforming value of $A_0[0]$ that satisfy Equation (19) by traversing a space of 2^c for $A_0[0]$. After that, all cells marked by "①" are determined.
2. In step ②, we have $D_0[3] = C_0[0] \oplus C_0[2]$ due to Equation (18), then we have

$$\begin{aligned}
&\Delta A_1[3] = \mathtt{S}(D_0[3]) \oplus \mathtt{S}(D_0'[3])\\
&= \mathtt{S}(C_0[0] \oplus C_0[2]) \oplus \mathtt{S}(C_0'[0] \oplus C_0'[2])\\
&= \mathtt{S}(\mathtt{S}(A_0[0]) \oplus K[0] \oplus \mathtt{S}(A_0[2])) \oplus \mathtt{S}(\mathtt{S}(A_0'[0]) \oplus K'[0] \oplus \mathtt{S}(A_0'[2]))\\
&= \mathtt{S}(\mathtt{S}(A_0[0]) \oplus K[0] \oplus \mathtt{S}(A_0[2])) \oplus \mathtt{S}(\mathtt{S}(A_0[0] \oplus \Delta A_0[0]) \oplus K'[0] \oplus \mathtt{S}(A_0[2] \oplus \Delta A_0[2])).
\end{aligned} \tag{20}$$

Since all the input-output differences of the super S-box and the pair of K are fixed, and $A_0[0]$ is determined by in step ①, only $A_0[2]$ is unfixed. We search $A_0[2]$ in a space of 2^8 to find the right one that make Equation (20) holds. All cells marked by "②" are fixed.

3. In step ③, we have $D_0[2] = C_0[1] \oplus C_0[2]$ due to Equation (18), then we have

$$\begin{aligned}
&\Delta A_1[2] = \mathtt{S}(D_0[2]) \oplus \mathtt{S}(D'_0[2])\\
&= \mathtt{S}(C_0[1] \oplus C_0[2]) \oplus \mathtt{S}(C'_0[1] \oplus C'_0[2])\\
&= \mathtt{S}(\mathtt{S}(A_0[1]) \oplus K[1] \oplus \mathtt{S}(A_0[2])) \oplus \mathtt{S}(\mathtt{S}(A'_0[1]) \oplus K'[1] \oplus \mathtt{S}(A'_0[2]))\\
&= \mathtt{S}(\mathtt{S}(A_0[1]) \oplus K[1] \oplus \mathtt{S}(A_0[2])) \oplus \mathtt{S}(\mathtt{S}(A_0[1] \oplus \Delta A_0[1]) \oplus K'[1] \oplus \mathtt{S}(A_0[2] \oplus \Delta A_0[2])).
\end{aligned} \tag{21}$$

Since all the input-output differences of the super S-box and the pair of K are fixed, and $A_0[2]$ is determined by in step ②, only $A_0[1]$ is unfixed. We search $A_0[1]$ in a space of 2^8 to find the right one that make Equation (21) holds. All cells marked by "③" are fixed.

4. In step ④, we have $D_0[0] = C_0[0] \oplus C_0[2] \oplus C_0[3]$ due to Equation (18), then we have

$$\begin{aligned}
&\Delta A_1[0] = \mathtt{S}(D_0[0]) \oplus \mathtt{S}(D'_0[0])\\
&= \mathtt{S}(C_0[0] \oplus C_0[2] \oplus C_0[3]) \oplus \mathtt{S}(C'_0[0] \oplus C'_0[2] \oplus C'_0[3])\\
&= \mathtt{S}(\mathtt{S}(A_0[0]) \oplus K[0] \oplus \mathtt{S}(A_0[2]) \oplus \mathtt{S}(A_0[3])) \oplus \mathtt{S}(\mathtt{S}(A'_0[0]) \oplus K'[0] \oplus \mathtt{S}(A'_0[2]) \oplus \mathtt{S}(A'_0[3]))\\
&= \mathtt{S}(\mathtt{S}(A_0[0]) \oplus K[0] \oplus \mathtt{S}(A_0[2]) \oplus \mathtt{S}(A_0[3])) \oplus\\
&\mathtt{S}(\mathtt{S}(A_0[0] \oplus \Delta A_0[0]) \oplus K'[0] \oplus \mathtt{S}(A_0[2] \oplus \Delta A_0[2]) \oplus \mathtt{S}(A_0[3] \oplus \Delta A_0[3])).
\end{aligned} \tag{22}$$

Since all the input-output differences of the super S-box and the pair of K are fixed, and $A_0[0]$ and $A_0[2]$ are already determined by in step ② and ③, only $A_0[3]$ is unfixed. We search $A_0[3]$ in a space of 2^8 to find the right one that make Equation (21) holds. All cells marked by "④" are fixed.

Following the above computing order, given an input-output difference $(\Delta A_0, \Delta A_1)$ with fixed key pair, we find the conforming pair for the full active super S-box in time complexity of about 2^8 two-round computations without any memory. Note that if the `MC` operation adopts MDS matrix, without memory, we need 2^{32} classical time to find a conforming pair for full active super S-box.

5.2 Collision on Hashing Modes with Reduced `SKINNY-128-384`

By applying the model given in Section 3, we find 16-round rebound trail for `SKINNY-128-384` as shown in Figure 19 in Supplementary Materials D. The inbound phase covers round 11 and round 12. The probability of the outbound phase is 2^{-112}. We apply similar technique of super S-box with non-MDS matrix to the inbound phase of the 16-round rebound trail. To be more clear, we list the details for solving the inbound phase in Supplementary Materials D.

For each super S-box, there are at most four spaces of size 2^8 to traverse to find the conforming pair, where 4 S-boxes are evaluated in each iteration (i.e., Equation (34), (35), (36) or (37)). Hence, in worst case, the time complexity to compute a super S-box is $4 \times 2^8 \times 4 = 2^{12}$ S-boxes evaluations. In quantum setting, we embed the Grover's algorithm to search the solutions. The quantum time to compute the one super S-box is (including uncomputing)

$$2 \cdot 4 \cdot \frac{\pi}{4} \cdot \sqrt{2^8} \cdot 4 = 2^{8.65} \quad \text{S-box evaluations,} \tag{23}$$

where in each Grover iteration, about 4 S-boxes are applied. Define $F : \mathbb{F}_2^{112} \times \mathbb{F}_2^3 \mapsto \mathbb{F}_2$ as $F(\Delta K, \Delta X_{11}, \Delta Y_{12}; x, \alpha)$, where $x = K \in \mathbb{F}_2^{112}$ and $\alpha = \alpha_0 \| \alpha_1 \| \alpha_2 \in \mathbb{F}_2^3$. $F(\Delta K, \Delta X_{11}, \Delta Y_{12}; x, \alpha) = 1$ if and only if $(\Delta K, \Delta X_{11}, \Delta Y_{12}; x, \alpha)$ leads a collision. The overall time complexity is

$$\frac{\pi}{4} \cdot \sqrt{2^{112+3}} \cdot 4 \cdot 2^{8.65} = 2^{67.8} \quad \text{S-box evaluations,} \tag{24}$$

which is about $2^{67.8}/256 = 2^{59.8}$ 16-round `SKINNY-128-384`, since there are 256 S-boxes in the 16-round `SKINNY-128-384`.

6 Free-Start Collision Attack on 9-Round `Whirlpool`

Different from `Saturnin` and `SKINNY`, the key schedule of `Whirlpool` is nonlinear. Hence, we have to tweak the automatic tool in Section 3 which targets on linear key schedule ciphers. For `Whirlpool`, we place the rebound attacks in both the encryption data path and the key schedule path just like Sasaki et al.'s work [48]. For the inbound part of the key schedule path, we only have input and output differences Δ_{in}^K and Δ_{out}^K that act as the degrees of freedom to preform the rebound attack in the key. We expect to get $|\Delta_{in}^K| \cdot |\Delta_{out}^K|$ key pairs conforming to the inbound part of key schedule path. For each key pair, we will get $|\Delta_{in}^E| \cdot |\Delta_{out}^E|$ pairs conforming to the inbound part of the encryption data path. Suppose the total probability of outbound paths in the key and encryption path is p, then the condition $|\Delta_{in}^K| \cdot |\Delta_{out}^K| \cdot |\Delta_{in}^E| \cdot |\Delta_{out}^E| \geq 1/p$ should be satisfied to finally find a key pair and data pair fulfilling the whole trails in the key schedule and the encryption data path. We embed the 2-round full/non-full active super S-box technique [20,47] or 3-round-inbound technique [29] in the inbound part. The inbound phase in related-key setting is quite similar to the single-key setting. A slight different point is to deal with the operation of XOR the key difference into the internal state, where the constraint [41] for truncated differential in XOR operation is applied. The outbound phase is also similar to single-key setting, where only propagations of truncated differential are constrained with MILP.

At ASIACRYPT 2012, Sasaki et al. [48] introduced a free-start collision attack on 8-round `Whirlpool`. In this section, we find a new 9-round rebound characteristic in Figure 13, and based on it, we give the quantum free-start collision on 9-round `Whirlpool`.

6.1 Comparison between Sasaki et al's Trail and Ours

The number of active S-boxes in Sasaki et al's 8-round trail is shown below:

$$\begin{cases} Key : 64 \xrightarrow{1^{st}R} 8 \xrightarrow{2^{nd}R} 1 \xrightarrow{3^{rd}R} 8 \xrightarrow{4^{th}R} 64 \xrightarrow{5^{th}R} 8 \xrightarrow{6^{th}R} 1 \xrightarrow{7^{th}R} 8 \xrightarrow{8^{th}R} 64, \\ Data : 0 \xrightarrow{1^{st}R} 8 \xrightarrow{2^{nd}R} 1 \xrightarrow{3^{rd}R} 8 \xrightarrow{4^{th}R} 0 \xrightarrow{5^{th}R} 8 \xrightarrow{6^{th}R} 1 \xrightarrow{7^{th}R} 8 \xrightarrow{8^{th}R} 64. \end{cases}$$

The number of active S-boxes in our 9-round trail is shown below:

$$\begin{cases} Key: 64 \xrightarrow{1^{st}R} 8 \xrightarrow{2^{nd}R} 1 \xrightarrow{3^{rd}R} 8 \xrightarrow{4^{th}R} 64 \xrightarrow{5^{th}R} 64 \xrightarrow{6^{th}R} 64 \xrightarrow{7^{th}R} 8 \xrightarrow{8^{th}R} 8 \xrightarrow{9^{th}R} 64, \\ Data: 0 \xrightarrow{1^{st}R} 8 \xrightarrow{2^{nd}R} 1 \xrightarrow{3^{rd}R} 0 \xrightarrow{4^{th}R} 64 \xrightarrow{5^{th}R} 64 \xrightarrow{6^{th}R} 64 \xrightarrow{7^{th}R} 8 \xrightarrow{8^{th}R} 8 \xrightarrow{9^{th}R} 64. \end{cases}$$

In the key schedule, Sasaki et al's inbound phase "$8 \xrightarrow{4^{th}R} 64 \xrightarrow{5^{th}R} 8$" is replaced by a longer inbound phase "$8 \xrightarrow{4^{th}R} 64 \xrightarrow{5^{th}R} 64 \xrightarrow{6^{th}R} 64 \xrightarrow{7^{th}R} 8$" in our trail, namely we gain a 2-round extension in the inbound phase. In the meantime, Sasaki et al.'s outbound part "$8 \xrightarrow{6^{th}R} 1 \xrightarrow{7^{th}R} 8 \xrightarrow{6^{th}R} 64$" is shortened to "$8 \xrightarrow{8^{th}R} 8 \xrightarrow{9^{th}R} 64$" to gain enough degrees of freedom. In Sasaki et al.'s 8-round trail, the full active state to match in the inbound phase only happens to the key schedule data path. In the inbound part of the encryption data path, many cells are inactive, so that one can assign arbitrary values. Hence, we do not worry about the degree of freedom for Sasaki et al.'s trail. However, in our 9-round trail, both the key and data path adopt full state active inbound part, so that the internal states are fully determined by a match-in-the-middle approach and the degree of freedoms only comes from the possible input and output differences of the inbound part. Hence, the outbound phase is different to gain enough degrees of freedom for the collision attack.

In the key schedule path, the inbound part covers from ΔB_3 to ΔC_6 that includes 3 `SB` layers. We apply Jean et al.'s [29] 3-round-inbound technique and their quantum version by Hosoyamada and Sasaki [25] to perform the attack. Following Section 2.3, we build the dedicated quantum algorithm for the 3-round inbound part with $d = 8, c = 8$ as shown in Figure 4. We first define G in Algorithm 4 which marks the compatible 🟥 cells in Figure 4 for (X_1, X_1') for a given input difference ΔX_1 and output difference ΔY_3.

Complexity of $\mathcal{U}_G$ in Algorithm 4 . Taken uncomputing into account, there are $32 \times 2 \times 2 \times 2 = 128$ S-boxes operations in Line 3. In Line 5 to Line 9, we need

$$8 \cdot \frac{\pi}{4} \cdot \sqrt{2^{64}} \cdot 16 \times 2 = 2^{39.65} \quad \text{S-boxes operations.} \tag{25}$$

In Line 11 of Algorithm 4, only the ■ cells are needed to compute backward to X_1, hence, $32 \times 2 \times 2 \times 2 = 128$ S-boxes operations are needed. Totally, we need about $2^{39.65}$ S-boxes operations to implement $\mathcal{U}_G$.

Given $(\Delta X_1, \Delta Y_3)$, run Grover's algorithm on $\mathcal{U}_G$ to find the correct 🟥 cells for (X_1, X_1') in Figure 4. $\mathcal{U}_G$ outputs 1 with probability of 2^{-256}. Hence, the time complexity to find the correct value with Grover's algorithm is

$$\frac{\pi}{4} \cdot \sqrt{2^{256}} \cdot 2^{39.65} = 2^{167.3} \quad \text{S-boxes operations.} \tag{26}$$

6.2 Free-Start Collision on 9-round `Whirlpool`

Classical Analysis on the 9-round Rebound Trail. As shown in Figure 13, in the key schedule part, given an input-output difference ΔB_3 and ΔC_6

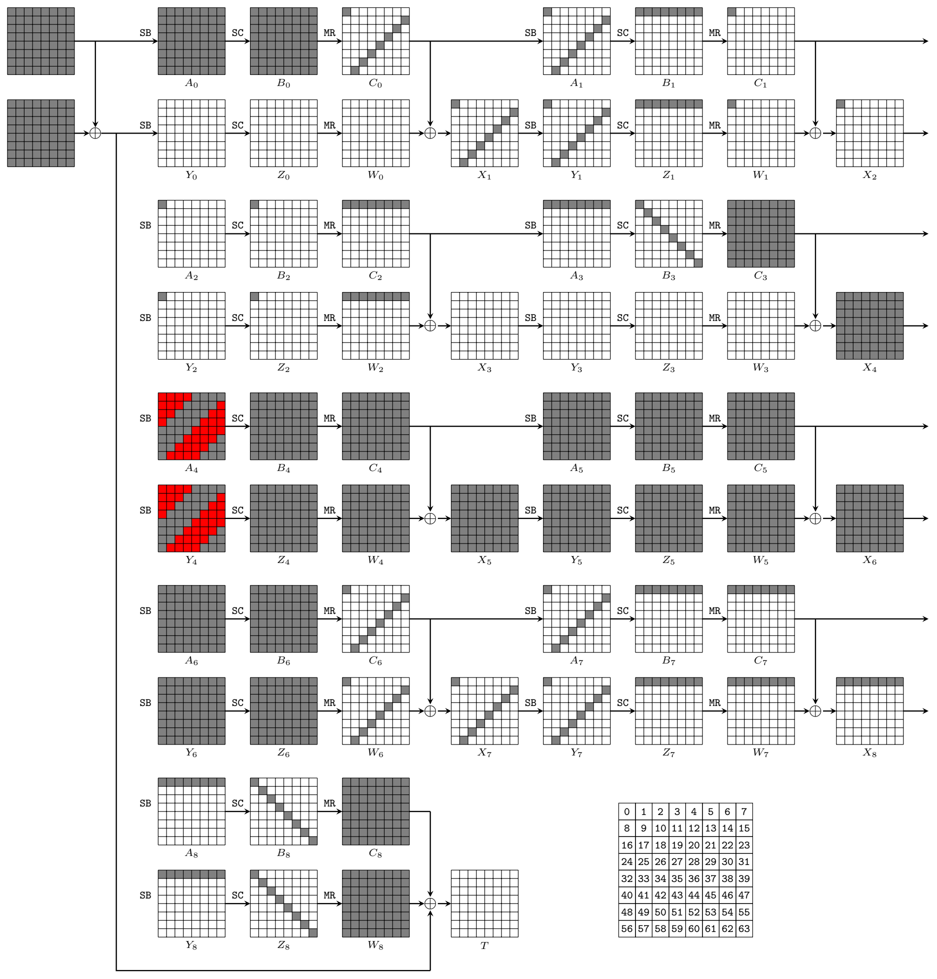

SB
SC
MR
A_0 B_0 C_0 A_1 B_1 C_1
Y_0 Z_0 W_0 X_1 Y_1 Z_1 W_1 X_2
A_2 B_2 C_2 A_3 B_3 C_3
Y_2 Z_2 W_2 X_3 Y_3 Z_3 W_3 X_4
A_4 B_4 C_4 A_5 B_5 C_5
Y_4 Z_4 W_4 X_5 Y_5 Z_5 W_5 X_6
A_6 B_6 C_6 A_7 B_7 C_7
Y_6 Z_6 W_6 X_7 Y_7 Z_7 W_7 X_8
A_8 B_8 C_8
Y_8 Z_8 W_8 T
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63

Fig. 13: Free-start collision attack on 9-round `Whirlpool`

Algorithm 4: Implementation of $\mathcal{U}_G$ without using qRAMs

```
   Input: $|\Delta X_1, \Delta Y_3; X_1[🟥]\rangle\,|y\rangle$
   Output: $|\Delta X_1, \Delta Y_3; X_1[🟥]\rangle\,|y \oplus G(\Delta X_1, \Delta Y_3; X_1[🟥])\rangle$
1  /* $X_1[🟥]$ means the value of 32 🟥 cells in state $X_1$ shown in Figure
     4, and $X_1'[🟥]$ is for state $X_1'$                                  */
2  Compute $X_1'[🟥] = \Delta X_1 \oplus X_1[🟥]$
3  Compute $Z_2[🟥]$ and $Z_2'[🟥]$ by $X_1[🟥]$ and $X_1'[🟥]$, respectively
4  Define $g_j : \mathbb{F}_2^{8\times 8} \mapsto \mathbb{F}_2$ for row $j = 0, 1, 2..., 7$ of $Z_2$. E.g., for row 0, define
     $g_0(Z_2[🟥], Z_2'[🟥]; x)$, where $x$ is the ■ cells of $Z_2$ and $Z_2'$ in row 0, i.e.,
     $x = Z_2[1,2,3,4] \| Z_2'[1,2,3,4] \in \mathbb{F}_2^{8\times 8}$. $g_0(Z_2[🟥], Z_2'[🟥]; x) = 1$ if and only if
     $\mathtt{SB}(\mathtt{MR}(Z_2[0,1,...,7])) \oplus \mathtt{SB}(\mathtt{MR}(Z_2'[0,1,...,7])) = \Delta Y_3[0,1,...,7]$. Similar
     property holds for other $g_j$
5  Run the Grover search on $g_0(Z_2[🟥], Z_2'[🟥]; \cdot) : \mathbb{F}_2^{8\times 8} \mapsto \mathbb{F}_2$. Let
     $Z_2[1,2,3,4] \| Z_2'[1,2,3,4]$ be the output.
6  Run the Grover search on $g_1(Z_2[🟥], Z_2'[🟥]; \cdot) : \mathbb{F}_2^{8\times 8} \mapsto \mathbb{F}_2$. Let
     $Z_2[10,11,12,13] \| Z_2'[10,11,12,13]$ be the output.
7  Run the Grover search on $g_2(Z_2[🟥], Z_2'[🟥]; \cdot) : \mathbb{F}_2^{8\times 8} \mapsto \mathbb{F}_2$. Let
     $Z_2[19,20,21,22] \| Z_2'[19,20,21,22]$ be the output.
8    $\vdots$
9  Run the Grover search on $g_7(Z_2[🟥], Z_2'[🟥]; \cdot) : \mathbb{F}_2^{8\times 8} \mapsto \mathbb{F}_2$. Let
     $Z_2[56,57,58,59] \| Z_2'[56,57,58,59]$ be the output.
10 /* Now the whole states $Z_2$ and $Z_2'$ are fixed.                      */
11 Compute backward from $Z_2$ and $Z_2'$ to $X_1$ and $X_1'$
12 if $X_1[■] \oplus X_1'[■] = \Delta X_1[■]$ then
13     return $|\Delta X_1, \Delta Y_3; X_1[🟥]\rangle\,|y \oplus 1\rangle$
14 else
15     return $|\Delta X_1, \Delta Y_3; X_1[🟥]\rangle\,|y\rangle$
```

of the inbound part, we call Algorithm 9 to find a conforming pair on average. In the outbound phase of the key schedule, the probability that the truncated differential ΔC_2 propogates to ΔB_2 is 2^{-56}. Hence, there will be $2^{64\times 2-56} = 2^{72}$ valid key pairs that meet the truncated differential in the key schedule path. For each valid key pair, we look at the encryption data path. ΔX_4 is fixed by ΔC_3, and there are 2^{64} possible differences in ΔW_6. There is also a 3-round inbound phase in the encryption data path with input difference ΔX_4 and output difference ΔW_6. With a given $(\Delta X_4,\ \Delta W_6)$, it is expected to find one data pair (X_4, X_4'). Together with the key pair, we compute backward with the data pair (X_4, X_4'). Since $\Delta W_2 = \Delta C_2$ and $\Delta B_2 = \mathtt{MR}^{-1}(\Delta C_2)$ whose row 0 is of $(*, 0, 0, 0, 0, 0, 0, 0)$, ΔZ_2 is also of the truncated form $(*, 0, 0, 0, 0, 0, 0, 0)$ with probability 1. At round 0, $\Delta W_0 = \Delta C_0 \oplus \Delta X_1 = 0$ holds with probability of 2^{-64}. At the last round, $\Delta B_8 = \Delta Z_8$ holds with probability 2^{-64}, which finally leads to a collision. The total degrees of freedom are derived from ΔB_3, ΔC_6 and ΔW_6, which consists of 2^{192} possible differences (24-byte). The probability to generate a collision is $2^{-56-128} = 2^{-184}$. To compute the 3-round inbound phase with Algorithm 9, the time complexity is about 2^{320}. Obviously, the classical complexity will be much larger that a generic birthday attack, which only needs 2^{256} time to find the 512-bit collision.

Quantum Free-Start Collision Attack on 9-round `Whirlpool`. In the key schedule path, for given $C_6[\mathbb{U}]$ with $\mathbb{U} = \{0, 15, 22, 29, 36, 43, 50, 57\}$ positions of active cells in C_6, we define $f : \mathbb{F}_2^{8\times8} \mapsto \mathbb{F}_2$ as $f(\Delta C_6[\mathbb{U}]; x)$, where $x = \Delta B_3[\mathbb{V}] \in \mathbb{F}_2^{8\times8}$ with $\mathbb{V} = \{0, 9, 18, 27, 36, 45, 54, 63\}$. $f(\Delta C_6[\mathbb{U}]; x) = 1$ if and only if the key pair derived by solving the 3-round inbound satisfies the truncated differential from ΔC_2 to ΔB_2. The implementation of $\mathcal{U}_f$ is given in Algorithm 5.

Complexity of $\mathcal{U}_f$. The time is bounded by Line 2 of Algorithm 5, which is about $2^{167.3}$ S-boxes operations according to Equation (26).

Algorithm 5: $\mathcal{U}_f$ of the quantum attack on 9-round `Whirlpool`

```
Input: $|\Delta C_6[\mathbb{U}]; \Delta B_3[\mathbb{V}]\rangle\,|y\rangle$
Output: $|\Delta C_6[\mathbb{U}]; \Delta B_3[\mathbb{V}]\rangle\,|y \oplus f(\Delta C_6[\mathbb{U}]; \Delta B_3[\mathbb{V}])\rangle$
1 Compute $\Delta C_3$ and $\Delta A_6$ from $\Delta B_3[\mathbb{V}]$ and $\Delta C_6[\mathbb{U}]$
2 Run Grover's algorithm on $G(\Delta C_3, \Delta A_6; \cdot)$ with $\mathcal{U}_G$ implemented in
    Algorithm 4. Let $A_4[■]$ be the output
3 /* $A_4[■]$ are the ■ cells in $A_4$ in Figure 13                    */
4 Run Line 2 to Line 11 of Algorithm 4 with input $(\Delta C_3, \Delta A_6; A_4[■])$. Let
    $(C_3, C_3')$ be the output
5 Compute backward from $(C_3, C_3')$ to $(B_2, B_2')$
6 if row 0 of $\Delta B$ is of the truncated form $(*, 0, 0, 0, 0, 0, 0, 0)$ then
7     return $|\Delta C_6[\mathbb{U}]; \Delta B_3[\mathbb{V}]\rangle\,|y \oplus 1\rangle$
8 else
9     return $|\Delta C_6[\mathbb{U}]; \Delta B_3[\mathbb{V}]\rangle\,|y\rangle$
```

Run Grover's algorithm on $f(\Delta C_6[\mathbb{U}]; \cdot)$, we will find a key pair (K, K') that conforms to the truncated differential in Figure 13. In encryption data path, for the computed key pair (K, K'), we define $\tilde{f} : \mathbb{F}_2^{8\times8} \mapsto \mathbb{F}_2$ as $\tilde{f}(K, K'; x)$, where $x = \Delta W_6[\mathbb{U}] \in \mathbb{F}_2^{8\times8}$. $\tilde{f}(K, K'; x) = 1$ if and only if a collision occurs in the digest that happens with probability of 2^{-128}. The implementation of $\mathcal{U}_{\tilde{f}}$ is given in Algorithm 6.

Complexity of $\mathcal{U}_{\tilde{f}}$. The time complexity is bounded by Line 3 of Algorithm 6, which is also $2^{167.3}$ S-boxes operations according to Equation (26).

We define $F : \mathbb{F}_2^{8\times8} \mapsto \mathbb{F}_2$ as $F(x)$, where $x = \Delta C_6[\mathbb{U}] \in \mathbb{F}_2^{8\times8}$ with $\mathbb{U} = \{0, 15, 22, 29, 36, 43, 50, 57\}$. $F(x) = 1$ if and only if the digests of two messages collide. The implementation of $\mathcal{U}_F$ is given in Algorithm 7.

Algorithm 6: $\mathcal{U}_{\tilde{f}}$ of the quantum attack on 9-round `Whirlpool`

```
Input: $|K, K'; \Delta W_6[\mathbb{U}]\rangle\,|y\rangle$
Output: $|K, K'; \Delta W_6[\mathbb{U}]\rangle\,|y \oplus \tilde{f}(K, K'; \Delta W_6[\mathbb{U}])\rangle$
1  Compute $\Delta X_4$ from $(K, K')$
2  Compute $\Delta Y_6$ from $\Delta W_6[\mathbb{U}]$
3  Run Grover's algorithm on $G(\Delta X_4, \Delta Y_6; \cdot)$ with $\mathcal{U}_G$ implemented in Algorithm
     4. Let $Y_4[🟥]$ be the output
4  /* $Y_4[🟥]$ are the 🟥 cells in $Y_4$ in Figure 13 */
5  Run Line 2 to Line 11 of Algorithm 4 with input $(\Delta X_4, \Delta Y_6, Y_4[🟥])$. Let
     $(X_4, X_4')$ be output
6  Together with $(K, K')$, compute backward from $(X_4, X_4')$ to $(X_1, X_1')$ and
     forward to $(W_8, W_8')$
7  Compute $(C_0, C_0')$ and $(C_8, C_8')$ by $(K, K')$
8  if $\Delta X_1 = \Delta C_0$ and $\Delta W_8 = \Delta C_8$ then
9      return $|K, K'; \Delta W_6[\mathbb{U}]\rangle\,|y \oplus 1\rangle$
10 else
11     return $|K, K'; \Delta W_6[\mathbb{U}]\rangle\,|y\rangle$
```

Complexity of $\mathcal{U}_F$. The time complexity of the implementation of $\mathcal{U}_F$ in Algorithm 7 is bounded by Line 1 and Line 4, which is

$$\frac{\pi}{4} \cdot \sqrt{2^{64-8}} \cdot 2^{167.3} + \frac{\pi}{4} \cdot \sqrt{2^{64}} \cdot 2^{167.3} = 2^{199.04} \text{ S-boxes operations.} \tag{27}$$

$\mathcal{U}_F$ returns $|\Delta C_6[\mathbb{U}]\rangle\,|y \oplus 1\rangle$ with probability of $2^{64-128} = 2^{-64}$. Hence, applying Grover's algorithm on $F(x)$ will finally find the collision. Since only the correct state $\Delta C_6[\mathbb{U}]$ is output, we have to re-run Line 1 to Line 6 of Algorithm 7 to finally find the collision. The total time complexity is bounded by the step of applying Grover's algorithm on $F(x)$, which is

$$\frac{\pi}{4} \cdot \sqrt{2^{64}} \cdot 2^{199.04} = 2^{230.7} \text{ S-boxes operations.} \tag{28}$$

Since there are $128 \times 9 = 1152$ S-boxes operations in the 9-round `Whirlpool`, the total time complexity of the quantum collision attack is $2^{230.7}/1152 = 2^{220.5}$ 9-round `Whirlpool`.

7 Conclusion

In this paper, by taking the degrees of freedom of the key materials into consideration, we build the automatic tools for the so-called related-key rebound attack, where the degrees of freedom are used to increase the probability of the outbound phase. We develop the new technique to deal with the incompatibilities when searching rebound-attack trails on `Saturnin`, whose subkeys have very strong relationships. Besides the automatic model, we build new super S-box technique

Algorithm 7: $\mathcal{U}_F$ of the quantum attack on 9-round `Whirlpool`

```
Input: $|\Delta C_6[\mathbb{U}]\rangle\,|y\rangle$
Output: $|\Delta C_6[\mathbb{U}]\rangle\,|y \oplus F(\Delta C_6[\mathbb{U}])\rangle$
1  Run Grover's algorithm on $f(\Delta C_6[\mathbb{U}];\cdot)$ with implementation of $\mathcal{U}_f$ in
     Algorithm 5. Let $\Delta B_3[\mathbb{V}]$ as output
2  /* Note that the truncated differential $\Delta C_2$ to $\Delta B_2$ holds with
      probability of $2^{-56}$, hence, about $\frac{\pi}{4}\sqrt{2^{56}}$ Grover iterations on
      $f(\Delta C_6[\mathbb{U}];\cdot)$ are needed to find a good one.                */
3  Run Line 1 to Line 5 of Algorithm 5 to get $(C_3, C_3')$, then compute the key
     pair $(K, K')$
4  Run Grover's algorithm on $\tilde{f}(K, K';\cdot)$ with implementation of $\mathcal{U}_{\tilde{f}}$ in
     Algorithm 6. Let $\Delta W_6[\mathbb{U}]$ as output
5  /* Note that since $\mathcal{U}_{\tilde{f}}$ returns 1 with probability of $2^{-128}$, however,
      the size of its domain is $2^{64}$. Then after $2^{32}$ Grover iterations,
      if a right $\Delta W_6[\mathbb{U}]$ is in the domain, then it will output. If all
      the $\Delta W_6[\mathbb{U}]$ are wrong, then a random $\Delta W_6[\mathbb{U}]$ will output.  */
6  Run Line 1 to Line 7 of Algorithm 6 to get $(X_1, X_1')$, then compute the
     message pair $(M, M')$ with $(K, K')$
7  if $(M, K)$'s digest collides with $(M', K')$'s then
8      return $|\Delta C_6[\mathbb{U}]\rangle\,|y \oplus 1\rangle$
9  else
10     return $|\Delta C_6[\mathbb{U}]\rangle\,|y\rangle$
```

with non-MDS matrix for `SKINNY`, which is not seen before. For `Whirlpool`, multiple nested Grover's algorithms are applied to deal with the complex case that both the key schedule path and encryption path adopt rebound attacks. All in all, we achieve certain best free-start collision attacks.

Acknowledgments

We would like to thank the reviewers for their valuable comments and suggestions. This work is supported by National Key R&D Program of China (2018YFA0704701, 2018YFA0704704), the Major Program of Guangdong Basic and Applied Research (2019B030302008), Major Scientific and Techological Innovation Project of Shandong Province, China (2019JZZY010133), Natural Science Foundation of China (61902207, 61772519, 62072270) and the Chinese Major Program of National Cryptography Development Foundation (MMJJ20180101, MMJJ20180102). The work is supported by the CAS Interdisciplinary Innovation Team.

References

1. Zhenzhen Bao, Jian Guo, Danping Shi, and Yi Tu. MITM meets guess-and-determine: Further improved preimage attacks against AES-like hashing. *IACR*

Cryptol. ePrint Arch., 2021:575, 2021.

2. Paulo S.L.M. Barreto and Vincent Rijmen. The WHIRLPOOL hashing function. *Submitted to NESSIE*, 2000.
3. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *CRYPTO 2016, Proceedings, Part II*, pages 123–153. Springer, 2016.
4. Daniel J. Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete. *SHARCS 2009 9: 105.*
5. Alex Biryukov and Ivica Nikolic. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, Camellia, Khazad and others. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110, pages 322–344. Springer, 2010.
6. Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. Quantum attacks without superposition queries: The offline Simon's algorithm. In *ASIACRYPT 2019, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, pages 552–583, 2019.
7. Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. Quantum security analysis of AES. *IACR Trans. Symmetric Cryptol.*, 2019(2):55–93, 2019.
8. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN '98, Campinas, Brazil, April, 20-24, 1998, Proceedings*, pages 163–169, 1998.
9. Anne Canteaut, Sébastien Duval, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Thomas Pornin, and André Schrottenloher. Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. *IACR Trans. Symmetric Cryptol.*, 2020(S1):160–207, 2020.
10. André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *ASIACRYPT 2017, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, pages 211–240, 2017.
11. Amit Kumar Chauhan, Abhishek Kumar, and Somitra Kumar Sanadhya. Quantum free-start collision attacks on double block length hashing with round-reduced AES-256. *IACR Trans. Symmetric Cryptol.*, 2021(1):316–336, 2021.
12. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. A security analysis of Deoxys and its internal tweakable block ciphers. *IACR Trans. Symmetric Cryptol.*, 2017(3):73–107, 2017.
13. Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO '89*, volume 435, pages 416–427.
14. Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815, pages 157–184.
15. Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881, pages 371–387.
16. Patrick Derbez, Paul Huynh, Virginie Lallemand, María Naya-Plasencia, Léo Perrin, and André Schrottenloher. Cryptanalysis results on Spook - bringing full-round Shadow-512 to the light. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172, pages 359–388. Springer.
17. Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks.

In *CRYPTO 2021, Proceedings, Part III*, volume 12827, pages 278–308. Springer, 2021.

18. Xiaoyang Dong, Siwei Sun, Danping Shi, Fei Gao, Xiaoyun Wang, and Lei Hu. Quantum collision attacks on AES-like hashing with low quantum random access memories. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020 Part II*, volume 12492, pages 727–757.
19. Pierre-Alain Fouque, Jérémy Jean, and Thomas Peyrin. Structural evaluation of AES and chosen-key distinguisher of 9-round AES-128. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042, pages 183–203. Springer.
20. Henri Gilbert and Thomas Peyrin. Super-Sbox cryptanalysis: Improved attacks for AES-like permutations. In *FSE 2010*, pages 365–383, 2010.
21. Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Physical Review A*, 78(5):052310, 2008.
22. Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.
23. Lorenzo Grassi, María Naya-Plasencia, and André Schrottenloher. Quantum algorithms for the k-xor problem. In *ASIACRYPT 2018*, pages 527–559, 2018.
24. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219, 1996.
25. Akinori Hosoyamada and Yu Sasaki. Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106, pages 249–279. Springer.
26. Akinori Hosoyamada and Yu Sasaki. Quantum collision attacks on reduced SHA-256 and SHA-512. In *CRYPTO 2021*, volume 12825, pages 616–646. Springer.
27. Akinori Hosoyamada and Yu Sasaki. Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In *CT-RSA 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, pages 198–218, 2018.
28. Akinori Hosoyamada and Yu Sasaki. Quantum Demiric-Selçuk Meet-in-the-Middle Attacks: Applications to 6-Round Generic Feistel Constructions. In *SCN 2018*, pages 386–403, 2018.
29. Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Improved rebound attack on the finalist Grøstl. In *FSE 2012*, pages 110–126, 2012.
30. Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Multiple limited-birthday distinguishers and applications. In *SAC 2013*, pages 533–550, 2013.
31. Jérémy Jean, María Naya-Plasencia, and Martin Schläffer. Improved analysis of ECHO-256. In *SAC*, pages 19–36, 2011.
32. Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Submission to CAESAR : Deoxys v1.41, October 2016. http://competitions.cr.yp.to/round3/deoxysv141.pdf.
33. Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In *CRYPTO 2016, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 207–237, 2016.
34. Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Quantum differential and linear cryptanalysis. *IACR Trans. Symmetric Cryptol.*, 2016(1):71–94, 2016.
35. Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In *ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 2682–2685, 2010.

36. Hidenori Kuwakado and Masakatu Morii. Security on the quantum-type Even-Mansour cipher. In *ISITA 2012, Honolulu, HI, USA, October 28-31, 2012*, pages 312–316, 2012.
37. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In *ASIACRYPT 2009*, pages 126–143.
38. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *FSE 2009*, pages 260–276.
39. Florian Mendel, Vincent Rijmen, and Martin Schläffer. Collision attack on 5 rounds of Grøstl. In *FSE 2014*, pages 509–521, 2014.
40. Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *CRYPTO '89*, volume 435, pages 428–446.
41. Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Inscrypt 2011*, volume 7537, pages 57–76, 2011.
42. María Naya-Plasencia. How to improve rebound attacks. In *CRYPTO 2011*, pages 188–205, 2011.
43. María Naya-Plasencia and André Schrottenloher. Optimal merging in quantum k-xor and k-xor-sum algorithms. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106, pages 311–340. Springer.
44. Boyu Ni, Xiaoyang Dong, Keting Jia, and Qidi You. (quantum) collision attacks on reduced simpira v2. *IACR Trans. Symmetric Cryptol.*, 2021(2):222–248, 2021.
45. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In Douglas R. Stinson, editor, *CRYPTO '93*, volume 773, pages 368–378.
46. Yu Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool. In *FSE 2011*, pages 378–396, 2011.
47. Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. Non-full-active Super-Sbox analysis: Applications to ECHO and Grøstl. In *ASIACRYPT 2010*, pages 38–55, 2010.
48. Yu Sasaki, Lei Wang, Shuang Wu, and Wenling Wu. Investigating fundamental security requirements on Whirlpool: Improved preimage and collision attacks. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658, pages 562–579.
49. Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptol.*, 12(1):1–28, 1999.

Supplementary Material

A Quantum Collision Attack on 7-round Saturnin

Our model also applies to single-key model by trivially let the key difference be 0. As shown in Figure 14, the inbound phase covers two SB layers from states Y_2 to Z_4. In the outbound phase, the probability of the truncated differential from ΔZ_1 to ΔY_1 and from ΔY_5 to ΔZ_5 are both 2^{-48}. The differences in plaintext and ciphertext collide with probability 2^{-64} when the truncated differential holds. Hence, the overall probability to find the collision is $2^{48\times 2+64} = 2^{-160}$, which can not be used in classical setting but works in quantum setting.

We need 2^{160} starting points computed by the inbound phase. However, the number of possible values for the input and output difference ΔY_2 and ΔZ_4 is only $2^{64\times 2} = 2^{128}$. Hence, we apply similar method by Dong et al. [18] that leverages two blocks to find the collision as shown in Figure 15. The rebound attack is placed in the second block. We need $2^{160-128} = 2^{32}$ m_0 to generate 2^{32} h_1 to build enough starting points in the second block. In the inbound phase, for a given h_1 and input-output difference $(\Delta Y_2, \Delta Z_4)$, we use the quantum version super S-boxes technique by Hosoyamada and Sasaki [25] to generate the conforming pairs. Similar to Hosoyamada and Sasaki's analysis [25], for given input-output difference $(\Delta Y_2, \Delta Z_4)$, we get (x, x') for each $\texttt{SSB}_i$ $(0 \leq i \leq 3)$ and we will obtain $(2 \cdot 2 \cdot 2 \cdot 2)/2 = 8$ starting points for each $(\Delta Y_2, \Delta Z_4)$. Hence, a 3-bit auxiliary variable is used to mark which to pick. Hence, we define $F(IV; m_0, \Delta Y_2, \Delta Z_4, \alpha) : \mathbb{F}_2^{163} \mapsto \mathbb{F}_2^1$, where $F(IV; m_0, \Delta Y_2, \Delta Z_4, \alpha) = 1$ if $(m_0, \Delta Y_2, \Delta Z_4, \alpha)$ leads a collision. Given $(m_0, \Delta Y_2, \Delta Z_4, \alpha)$, we apply the four embedded Grover's algorithm to solve the SSB defined by Hosoyamada and Sasaki's analysis [25]. The overall attack is quite similar to Hosoyamada and Sasaki's analysis [25] on 7-round AES without qRAM. At last, we need about $2^{163/2} \times 2^{64/2} = 2^{113.5}$ 7-round Saturnin-hash to build the collision attack without qRAM.

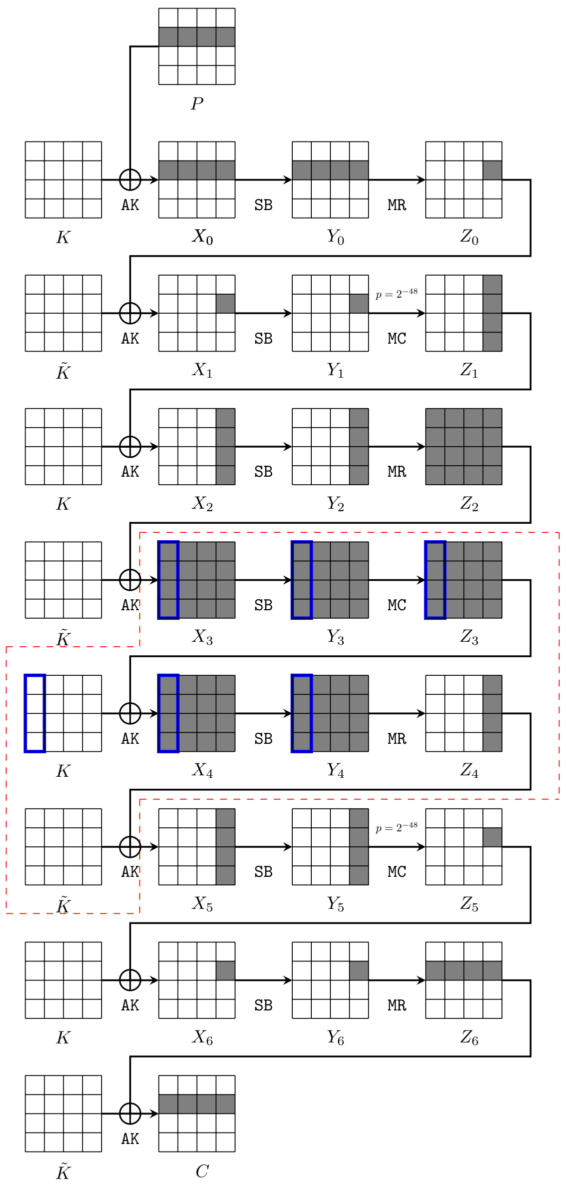


Fig. 14: Rebound Attack Trail for the Collision attack on 7-round `Saturnin`

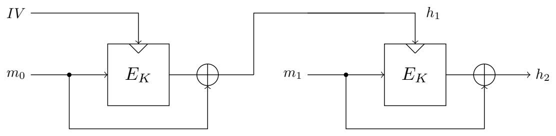


Fig. 15: Two blocks to build the collision on 7-round `Saturnin`

B Classic Collision Attack on Reduced Saturnin

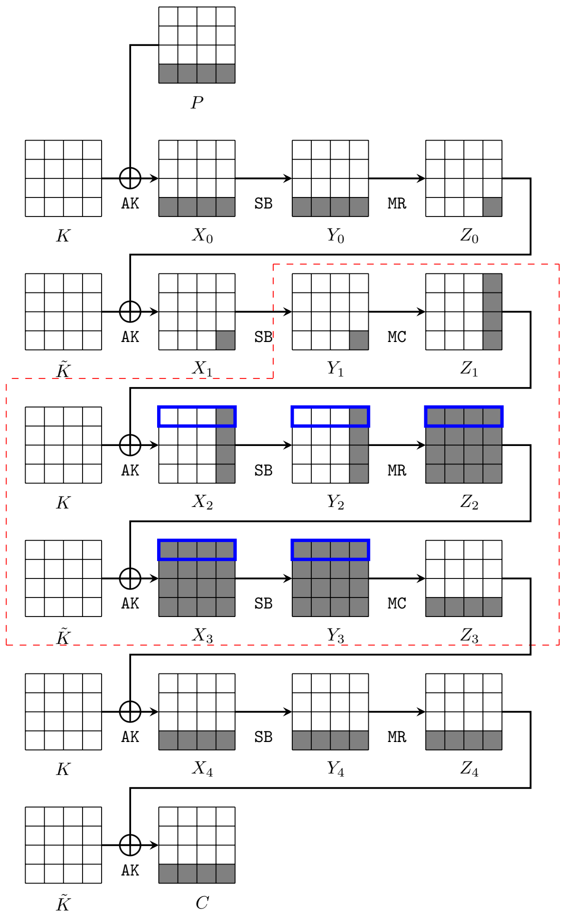


Fig. 16: Classic collision attack on 5-round Saturnin

When searching trails for classic rebound attack on Saturnin, we just tweak the target of the model in Section 3 by removing the factor of the complexity of inbound part. Namely, the new target for $r_{in} = 2$ is

$$\sum Prob_r^i + \sum x_0^{i,j}. \tag{29}$$

As discussed in [25] and also in Section 6.2, when $r_{in} = 3$, the classic time complexity to solve Jean et al's 3-round inbound part is too large to be useful in the classic rebound attack. Hence, we only consider the case $r_{in} = 2$. The probability of the outbound phase has to be larger than $2^{-n/2}$.

Classic collision on 5-round `Saturnin`. We find a 5-round trail as shown in Figure 16. The probability to collide the plaintext and ciphertext is 2^{-64}. The inbound part covers 2 round from state Y_1 to state Z_3. The attack procedures are:

1. For fixed ΔY_1, and compute the ΔX_2.
2. For each 64-bit value $X_2[0,4,8,12]$, compute $X_2'[0,4,8,12]$, $Y_3[0,4,8,12]$ and $Y_3'[0,4,8,12]$. Insert $X_2[0,4,8,12]$ into table $L_0[\Delta Y_3[0,4,8,12]]$.
3. Similarly, build table L_1, L_2, L_3.
4. For each ΔZ_3,
   (a) Compute ΔY_3 and access L_0, L_1, L_2, L_3 to get the pair (X_2, X_2').
   (b) Check if the pair (X_2, X_2') leads to a collision.

Since in Step 4, we have 2^{64} possible differences for ΔZ_3, we are expected to check 2^{64} (X_2, X_2'). Since the probability of the outbound phase is 2^{-64}, we are expected to get one collision. The time complexity is about 2^{64} and the memory is about $2^{64} \times 4 = 2^{66}$.

Classic free-start collision on 6-round `Saturnin`. As shown in Figure 17, the inbound phase covers two rounds from state Z_1 to Z_3. The trail in Figure 17 is much easier than Figure 10 and there are no conditions on the key. Hence, we just randomly pick a difference for the key ΔK and a key pair (K, K') with $K \oplus K' = \Delta K$ to perform the rebound attack. Then, the probability of the outbound phase is $2^{-16-64} = 2^{-80}$ (note that the difference in both plaintext and ciphertext is equal to the difference in K). The procedures are:

1. Chosen a fixed difference for the key, i.e., ΔK and a fixed pair (K, K').
2. For each ΔY_1, compute ΔX_2,
   (a) Build super S-box tables L_0, L_1, L_2, L_3.
   (b) For each ΔX_4,
      i. Compute ΔY_3,
      ii. Access L_0, L_1, L_2, L_3 to get the the pair (X_2, X_2'),
      iii. Check if (X_2, X_2') leads to a collision.

We have $2^{16+64} = 2^{80}$ possible differences for $(\Delta Y_1,\ \Delta X_4)$. Since the probability of the outbound phase is 2^{-80}, we are expected to find one collision. The time complexity is about 2^{80} and the memory complexity is 2^{66}.

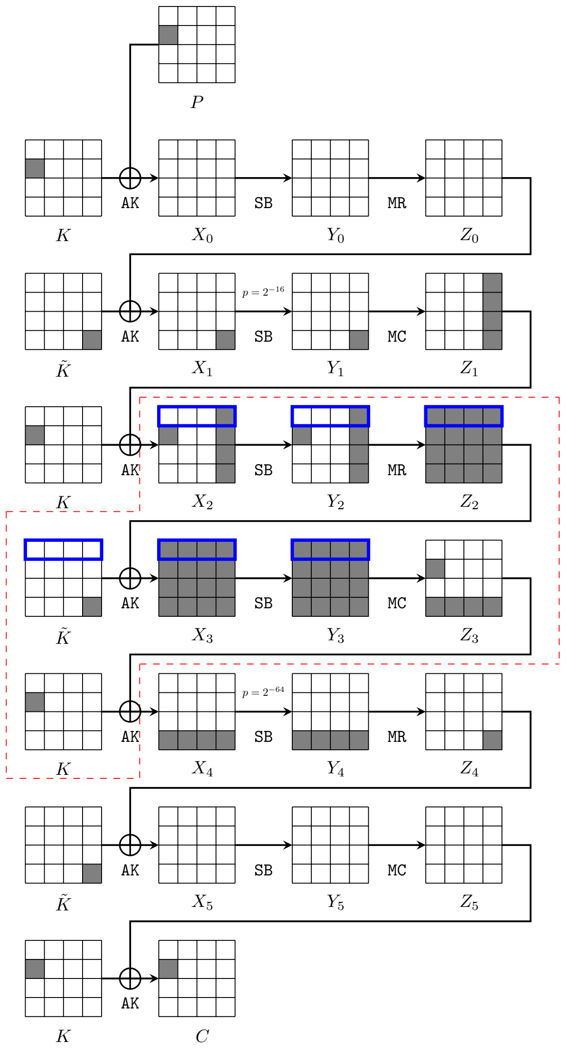


Fig. 17: Classic free-start collision attack on 6-round `Saturnin`

C Semi-free Start Collision Attack on 7-round `Saturnin`

In this section, we tweak the trail of the 7-round quantum collision in Figure 14 by extending the inbound phase, we get Figure 18 to perform a quantum semi-free-start collision attack on 7-round `Saturnin`. The inbound phase covers $\varDelta Y_2$ to $\varDelta Z_5$. The probability of the outbound phase (to lead a collision) is $2^{-64-48} = 2^{-112}$. Our quantum attack needs about $2^{90.99}$ time complexity without qRAM or classical memory, which is better than CNS algorithm.

Note that, as shown in Figure 10 and according to the key schedule of `Saturnin`, the last column of $\tilde{K}$ is only related to the first column of K. Denote $K^{(1,2,3)}$ as the 2nd, 3rd, 4th column of K and the first column as $K^{(0)}$.

We define $F(\varDelta Y_2, \varDelta Z_4, \varDelta Z_5; K^{(1,2,3)}, \alpha, \beta) : \mathbb{F}_2^{192} \times \mathbb{F}_2^3 \times \mathbb{F}_2^4 \mapsto \mathbb{F}_2^1$, where $K^{(1,2,3)} \in \mathbb{F}_2^{192}$, $\alpha \in \mathbb{F}_2^3$, $\beta \in \mathbb{F}_2^4$. $F(\varDelta Y_2, \varDelta Z_4, \varDelta Z_5; K^{(1,2,3)}, \alpha, \beta) = 1$ if and only if we get a collision with the input $(\varDelta Y_2, \varDelta Z_4, \varDelta Z_5; K^{(1,2,3)}, \alpha, \beta)$. The implementation of $\mathcal{U}_F$ is given in Algorithm 8.

Complexity of $\mathcal{U}_F$. According to Hosoyamada and Sasaki's method [25] of computing super S-box quantumly, the complexity of Line 6 Line 8 is

$$3 \cdot \frac{\pi}{4} \cdot \sqrt{2^{64}} \cdot 32 = 2^{38.24} \quad \text{S-box applications.} \tag{30}$$

The complexity of Line 12 to Line 15 is

$$4 \cdot \frac{\pi}{4} \cdot \sqrt{2^{16}} \cdot 4 = 2^{11.65} \quad \text{S-box applications.} \tag{31}$$

The complexity of Line 18 is

$$\frac{\pi}{4} \cdot \sqrt{2^{64}} \cdot 32 = 2^{36.65} \quad \text{S-box applications.} \tag{32}$$

Totally, the complexity to implement $\mathcal{U}_F$ is about $2^{38.24} + 2^{11.65} + 2^{36.65} = 2^{38.65}$ S-box applications.

Since the inbound phase covers $\varDelta Y_2$ to $\varDelta Z_5$, the probability of the outbound phase is $2^{-64-48} = 2^{-112}$. Hence, the success probability of $\mathcal{U}_F$ is $2^{-112-3-4} = 2^{-119}$ with 3-bit α and 4-bit β auxiliary variables. Run Grover's algorithm on $\mathcal{U}_F$ to get the collision, we have time complexity

$$\frac{\pi}{4} \cdot \sqrt{2^{119}} \cdot 2^{38.65} = 2^{97.8} \quad \text{S-box applications.} \tag{33}$$

Since we have $16 \times 7 = 112$ S-boxes for 7-round `Saturnin`, the complexity to perform the quantum semi-free-start collision attack is $2^{97.8}/112 = 2^{90.99}$ 7-round `Saturnin` without any memory, which is better than CNS algorithm (time $= 2^{102.4}$, classical memory $= 2^{51.2}$).

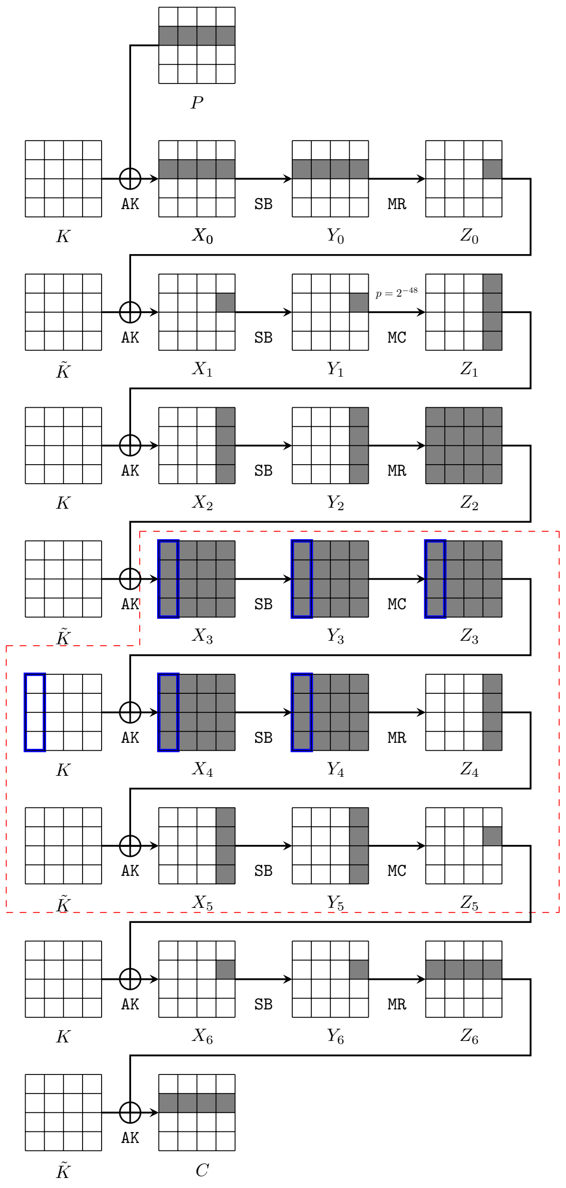


Fig. 18: Quantum semi-free-start collision attack on 7-round `Saturnin`

Algorithm 8: Implementation of $\mathcal{U}_F$ for 7-round semi-free-start collision on `Saturnin`

```
Input: $|\Delta Y_2, \Delta Z_4, \Delta Z_5; K^{(1,2,3)}, \alpha, \beta\rangle\,|y\rangle$ with $\alpha = (\alpha_1, \alpha_2, \alpha_3) \in \mathbb{F}_2^3$,
       $\beta = (\beta_0, \beta_1, \beta_2, \beta_3) \in \mathbb{F}_2^4$
Output: $|\Delta Y_2, \Delta Z_4, \Delta Z_5; K^{(1,2,3)}, \alpha, \beta\rangle\,|y \oplus F(\Delta Y_2, \Delta Z_4, \Delta Z_5; K^{(1,2,3)}, \alpha, \beta\rangle$
1  Compute $\Delta X_3$ from $\Delta Y_2$
2  Compute $\Delta Y_5$ from $\Delta Z_5$
3  Compute $\Delta Y_4$ from $\Delta Z_4$
4  /* Compute conforming pairs of $X_3$ for 2nd, 3rd, 4th super S-boxes
      with quantum memoryless method by Hosoyamada and Sasaki [25]  */
5  Let $G^{(i)} : \mathbb{F}_2^{64} \mapsto \mathbb{F}_2^1$ be a Boolean function such that
     $G^{(i)}(\Delta X_3^{(i)}, \Delta Y_4^{(i)}, \alpha_i = 0; X_3^{(i)}) = 1$ if and only if
     $\mathtt{SSB}^{(i)}(X_3^{(i)}) \oplus \mathtt{SSB}^{(i)}(X_3^{(i)} \oplus \Delta X_3^{(i)}) = \Delta Y_4^{(i)}$ and $X_3^{(i)} \leq X_3^{(i)} \oplus \Delta X_3^{(i)}$;
     $G^{(i)}(\Delta X_3^{(i)}, \Delta Y_4^{(i)}, \alpha_i = 1; X_3^{(i)}) = 1$ if and only if
     $\mathtt{SSB}^{(i)}(X_3^{(i)}) \oplus \mathtt{SSB}^{(i)}(X_3^{(i)} \oplus \Delta X_3^{(i)}) = \Delta Y_4^{(i)}$ and $X_3^{(i)} > X_3^{(i)} \oplus \Delta X_3^{(i)}$
6  Run Grover's algorithm on $G^{(1)}(\Delta X_3^{(1)}, \Delta Y_4^{(1)}, \alpha_1; \cdot) : \mathbb{F}_2^{64} \mapsto \mathbb{F}_2^1$. Let $X_3^{(1)}$ be
     the output
7  Run Grover's algorithm on $G^{(2)}(\Delta X_3^{(2)}, \Delta Y_4^{(2)}, \alpha_2; \cdot) : \mathbb{F}_2^{64} \mapsto \mathbb{F}_2^1$. Let $X_3^{(2)}$ be
     the output
8  Run Grover's algorithm on $G^{(3)}(\Delta X_3^{(3)}, \Delta Y_4^{(3)}, \alpha_3; \cdot) : \mathbb{F}_2^{64} \mapsto \mathbb{F}_2^1$. Let $X_3^{(3)}$ be
     the output
9  Deduce $Y_4^{(1,2,3)}$ from $X_3^{(1,2,3)}$
10 /* With $\Delta X_5 = \Delta Z_4$ and $\Delta Y_5$, we compute the pair $(X_5^{(3)}, X_5'^{(3)})$  */
11 Let $g_j : \mathbb{F}_2^{16} \to \mathbb{F}_2^1$ be a Boolean function such that $g_j(\delta_{in}, \delta_{out}, \beta_j = 0; x) = 1$ if
     and only if $S(x) \oplus S(x \oplus \delta_{in}) = \delta_{out}$ and $x \leq x \oplus \delta_{in}$, and
     $g_j(\delta_{in}, \delta_{out}, \beta_j = 1; x) = 1$ if and only if $S(x) \oplus S(x \oplus \delta_{in}) = \delta_{out}$, and
     $x > x \oplus \delta_{in}$.
12 Run Grover's algorithm on $g_0(\Delta X_5[12], \Delta Y_5[12], \beta_0; \cdot) : \mathbb{F}_2^{16} \mapsto \mathbb{F}_2^1$. Let $X_5[12]$
     be the output
13 Run Grover's algorithm on $g_1(\Delta X_5[13], \Delta Y_5[13], \beta_1; \cdot) : \mathbb{F}_2^{16} \mapsto \mathbb{F}_2^1$. Let $X_5[13]$
     be the output
14 Run Grover's algorithm on $g_2(\Delta X_5[14], \Delta Y_5[14], \beta_2; \cdot) : \mathbb{F}_2^{16} \mapsto \mathbb{F}_2^1$. Let $X_5[14]$
     be the output
15 Run Grover's algorithm on $g_3(\Delta X_5[15], \Delta Y_5[15], \beta_3; \cdot) : \mathbb{F}_2^{16} \mapsto \mathbb{F}_2^1$. Let $X_5[15]$
     be the output
16 /* After guessing $\tilde{K}^{(3)}$, we can compute $Z_4^{(3)}$. Together with $Y_4^{(1,2,3)}$
      and Property 1, we get $Y_4^{(0)}$. Moreover, we deduce $K^{(0)}$ from $\tilde{K}^{(3)}$.
      Then, $X_3^{(0)}$ is computed. We check if the computed $\Delta X_3^{(0)}$ is equal
      to that given in Line 1  */
17 Let $f : \mathbb{F}_2^{64} \mapsto \mathbb{F}_2^1$ be a Boolean function such that
     $f(\Delta X_3^{(0)}, \Delta Z_4^{(3)}, Y_4^{(1,2,3)}, X_5^{(3)}; \tilde{K}^{(3)}) = 1$ if and only if the computed $\Delta X_3^{(0)}$ is
     equal to that given in Line 1
18 Run Grover's algorithm on $f(\Delta X_3^{(0)}, \Delta Z_4^{(3)}, Y_4^{(1,2,3)}, X_5^{(3)}; \cdot) : \mathbb{F}_2^{64} \mapsto \mathbb{F}_2^1$. Let
     $\tilde{K}^{(3)}$ be the output
19 /* Together with $K^{(1,2,3)}$, the full state of $K$ is fixed  */
20 Compute $X_3^{(0)}$ with $\tilde{K}^{(3)}$
21 /* Now, we get one starting point $(X_3, X_3')$, which already fullfils
      the trail from $\Delta Y_2$ to $\Delta Z_5$ with the computed $K$  */
22 Compute backward and forward to the plaintext $(P, P')$ and ciphertext $(C, C')$
23 if $(P, P')$ and $(C, C')$ lead to a collision then
24     return $|\Delta Y_2, \Delta Z_4, \Delta Z_5; K^{(1,2,3)}, \alpha, \beta\rangle\,|y \oplus 1\rangle$
25 else
26     return $|\Delta Y_2, \Delta Z_4, \Delta Z_5; K^{(1,2,3)}, \alpha, \beta\rangle\,|y\rangle$
```

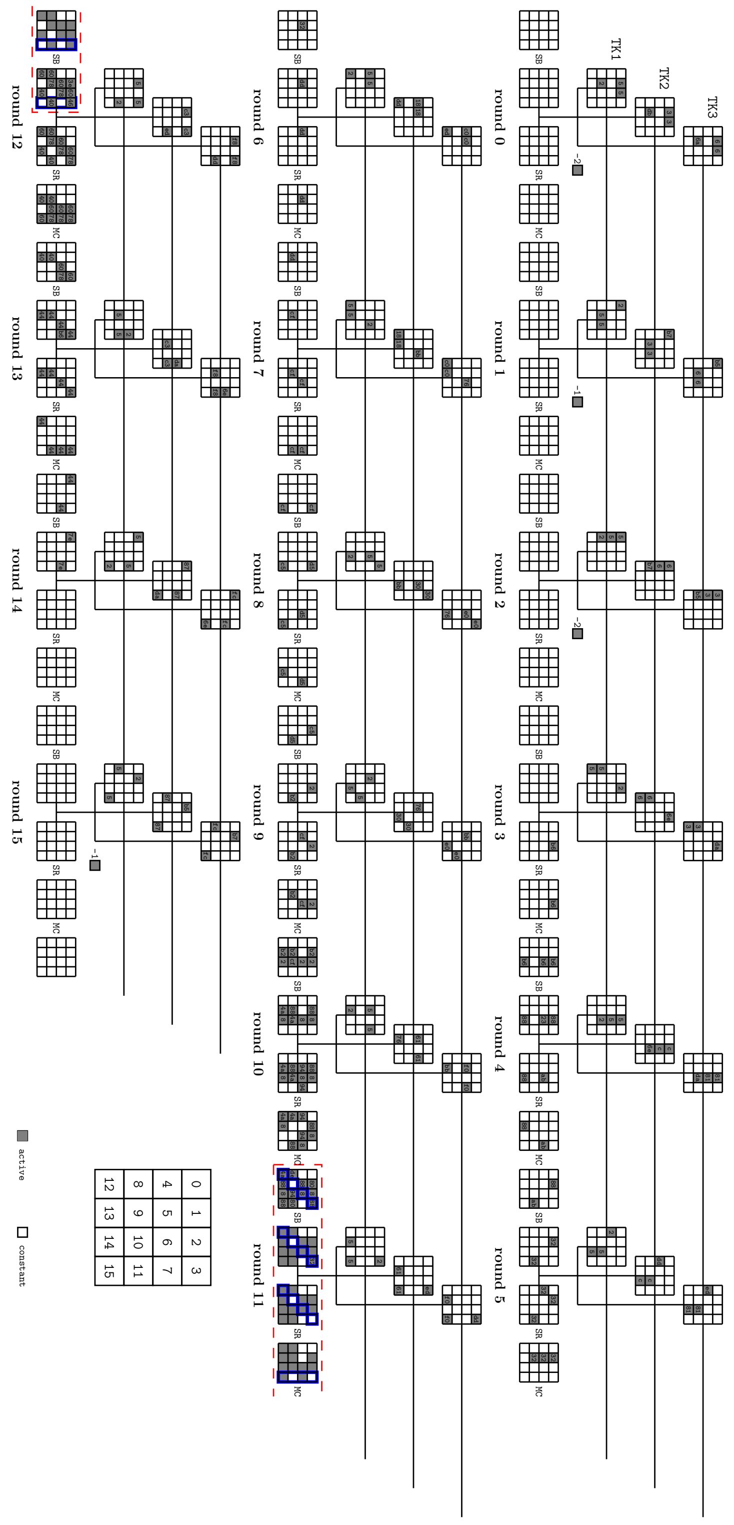


Fig. 19: Rebound Attack trail on 16-round SKINNY-128-384

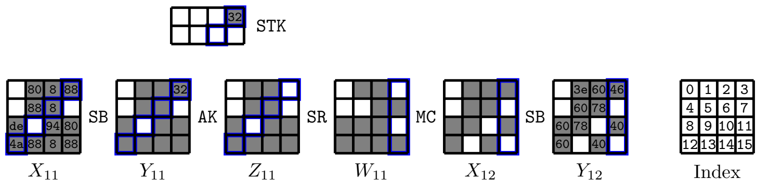


Fig. 20: Inbound part of the rebound attack on SKINNY

D Solving the Inbound Part of SKINNY

Now we solve the inbound part of the 16-round rebound trail in Figure 19. To be more clear, we pick out the inbound part in Figure 20. Since the inbound part are not fully active, we tweak some procedures for solving the super S-box to achieve better time complexity. For given key pair, we solve the inbound part (Figure 20) to get a starting point for the outbound part. Since the probability of the outbound part is 2^{-112}, we have to traverse 2^{112} key pairs.

Precomputation. We first precompute the solution for $\mathtt{S}(x \oplus \Delta X_{11}[3]) \oplus \mathtt{S}(x) = \Delta Y_{11}[3]$, where $\Delta X_{11}[3] = \mathtt{0x88}$ and $\Delta Y_{11}[3] = \mathtt{0x32}$. We random choose a value for the inactive cell $X_{11}[9]$ and compute $W_{11}[11] = Y_{11}[9] = \mathtt{S}(X_{11}[9])$. This step is precomputed and $X_{11}[3]$, $X_{11}[9]$, $W_{11}[11]$, $Y_{11}[9] = \mathtt{S}(X_{11}[9])$, $Y_{11}[3] = \mathtt{S}(X_{11}[3])$ and $Y'_{11}[3] = \mathtt{S}(X_{11}[3] \oplus \mathtt{0x88})$ are fixed for all the 2^{112} key pairs and through all the steps of the rebound attack.

For each key pair, we compute the inbound part as follows.

Computing the $\mathtt{SSB}_3$. For $\mathtt{SSB}_3$, (marked by blue boxes in Figure 20), due to Equation (18), we have $\Delta X_{12}[11] = \Delta W_{11}[7] \oplus \Delta W_{11}[11] = \Delta W_{11}[7]$, and $\mathtt{S}(X_{11}[6]) \oplus \mathtt{S}(X_{11}[6] \oplus \mathtt{0x8}) = \Delta W_{11}[7]$.

$$\begin{aligned}&\Delta Y_{12}[11] = \mathtt{0x40} = \mathtt{S}(X_{12}[11]) \oplus \mathtt{S}(X'_{12}[11])\\&= \mathtt{S}(W_{11}[7] \oplus W_{11}[11]) \oplus \mathtt{S}(W'_{11}[7] \oplus W'_{11}[11])\\&= \mathtt{S}(\mathtt{S}(X_{11}[6]) \oplus STK[6] \oplus Y_{11}[9]) \oplus \mathtt{S}(\mathtt{S}(X_{11}[6] \oplus \mathtt{0x8}) \oplus STK'[6] \oplus Y_{11}[9]),\end{aligned} \tag{34}$$

where only $X_{11}[6]$ is unfixed, $Y_{11}[9]$ and the STK are fixed. Hence, by traversing $X_{11}[6]$ with four S-boxes evaluations in each iteration, we get one value of $X_{11}[6]$ that meets the Equation (34).

Due to MC of SKINNY, we have $\Delta X_{12}[3] = \Delta W_{11}[15]$, since $\Delta W_{11}[3] = 0$ and $\Delta W_{11}[11] = 0$. Similar to Equation (34), we get

$$\begin{aligned}&\Delta Y_{12}[3] = \mathtt{0x46} = \mathtt{S}(X_{12}[3]) \oplus \mathtt{S}(X'_{12}[3])\\&= \mathtt{S}(W_{11}[3] \oplus W_{11}[11] \oplus W_{11}[15]) \oplus \mathtt{S}(W'_{11}[3] \oplus W'_{11}[11] \oplus W'_{11}[15])\\&= \mathtt{S}(Y_{11}[3] \oplus STK[3] \oplus Y_{11}[9] \oplus \mathtt{S}(X_{11}[12])) \oplus\\&\mathtt{S}(Y'_{11}[3] \oplus STK'[3] \oplus Y_{11}[9] \oplus \mathtt{S}(X_{11}[12] \oplus \mathtt{0x4a})),\end{aligned} \tag{35}$$

where $Y_{11}[3]$, $Y'_{11}[3]$, $Y_{11}[9]$ are pre-fixed, and only $X_{11}[12]$ is the variable. Hence, by traversing $X_{11}[12]$ with four S-boxes evaluations in each iteration, we are expected to find one $X_{11}[12]$ that meets the Equation (35).

Computing $\texttt{SSB}_2$. The input and output column $(W_{11}^{(2)}, X_{12}^{(2)})$ of $\texttt{MC}$ have four active cells and and three active cells. With the property of $\texttt{MC}$ by Equation (18), we know $X_{12}[6] = W_{11}[2]$. Then, we get

$$\begin{aligned}
&\Delta Y_{12}[6] = \texttt{0x78} = \texttt{S}(X_{12}[6]) \oplus \texttt{S}(X'_{12}[6]) \\
&= \texttt{S}(W_{11}[2]) \oplus \texttt{S}(W'_{11}[2]) \\
&= \texttt{S}(\texttt{S}(X_{11}[2]) \oplus STK[2]) \oplus \texttt{S}(\texttt{S}(X_{11}[2] \oplus \texttt{0x8}) \oplus STK'[2]),
\end{aligned} \tag{36}$$

where only $X_{11}[2]$ are unfixed, hence, by traversing $X_{11}[2]$, we get a value of $X_{11}[2]$ to meet the Equation (36). After $X_{11}[2]$ is determined, we compute $Y_{11}[2] = \texttt{S}(X_{11}[2])$ and $Y'_{11}[2] = \texttt{S}(X_{11}[2] \oplus \texttt{0x8})$, then we have

$$\begin{aligned}
&\Delta Y_{12}[14] = \texttt{0x40} = \texttt{S}(X_{12}[14]) \oplus \texttt{S}(X'_{12}[14]) \\
&= \texttt{S}(W_{11}[2] \oplus W_{11}[10]) \oplus \texttt{S}(W'_{11}[2] \oplus W'_{11}[10]) \\
&= \texttt{S}(Y_{11}[2] \oplus STK[2] \oplus \texttt{S}(X_{11}[8])) \oplus \texttt{S}(Y'_{11}[2] \oplus STK'[2] \oplus \texttt{S}(X_{11}[8] \oplus \texttt{0xde}))
\end{aligned} \tag{37}$$

where $Y_{11}[2]$ and $Y'_{11}[2]$ are fixed. Hence, the only unfixed cell is $X_{11}[8]$, by traversing it, we find a value $X_{11}[8]$ to meet the Equation (37). Similarly, we determine $X_{11}[5]$ and $X_{11}[15]$ by traversing a space of 2^8, respectively.

The computing of $\texttt{SSB}_0$ and $\texttt{SSB}_1$ is very similar to the above procedures in computing $\texttt{SSB}_2$ and $\texttt{SSB}_3$, which only traverses some small space of 2^8. For each super S-box, there are at most four spaces of size 2^8 to traverse to find the conforming pair, where 4 S-boxes are evaluated in each iteration (i.e., Equation (34), (35), (36) or (37)). Hence, in worst case, the time complexity to compute a super S-box is $4 \times 2^8 \times 4 = 2^{12}$ S-boxes evaluations. In quantum setting, we embed the Grover's algorithm to search the solutions similar to Dong et al.'s [18] in computing the non-full-active super S-box. The quantum time to compute the one super S-box is

$$2 \cdot 4 \cdot \frac{\pi}{4} \cdot \sqrt{2^8} \cdot 4 = 2^{8.65} \quad \text{S-box evaluations,} \tag{38}$$

where in each Grover iteration, about 4 S-boxes are applied, e.g. Equation (34), (35), (36) or (37), and uncomputing is considered.

E Memoryless Method to Solve the 3-Round Inbound Part

Suppose the state is of $d \times d$ c-bit cells. As shown in Figure 4, ΔX_1 is linearly computed from ΔZ_0. Similar to super S-box technique, one computes d super S-boxes $\texttt{SSB}_j^f$ $(0 \le j < d)$ covering X_1 to Y_2. Each $\texttt{SSB}_j^f$ contains 2^{dc} pairs with fixed input differences. Similarly, the attacker computes d inverse super S-boxes $\mathtt{SSB}_j^b$ $(0 \le j < d)$ from Y_3 to Y_2.

Match $2d$ lists by the exhaustive search of $\mathtt{SSB}_0^f, \mathtt{SSB}_1^f, ..., \mathtt{SSB}_{d/2-1}^f$. Then, a pair of $d^2/2$ cells at Y_2 are fixed. The attacker then checks if the pair of fixed values can be produced by $\mathtt{SSB}_j^b$. In each $\mathtt{SSB}_j^b$, d-cell values have been fixed, which acts as a dc-bit filter. Since the degrees of freedom in each $\mathtt{SSB}_j^b$ is 2^{dc}, there expects one match on average for each $\mathtt{SSB}_j^b$. Now, the pair at Y_2 is fully fixed. Then, compute the pair backward to X_1 to verify if the pair conforms to the fixed difference ΔX_1, which acts as a filter of $2^{-d^2c/2}$. Hence, by the exhaustive search of the $2^{d^2c/2}$ values of $\mathtt{SSB}_0^f, \mathtt{SSB}_1^f, ..., \mathtt{SSB}_{d/2-1}^f$, one expects to find a solution for the 3-round inbound phase, which conforms to the fixed input and output differences.

At EUROCRYPT 2020, Hosoyamada and Sasaki [25] introduced a memoryless variant as shown in Algorithm 9 , and then converted it into a quantum variant. The time complexity of Algorithm 9 is $2^{t^2c/2+tc}$ and there expect one conforming pair as output.

Algorithm 9: Memoryless variant of the 3-round inbound phase

```
    Input: $\Delta Z_0$ and $\Delta W_3$
    Output: $(Z_0, Z_0')$ and $(W_3, W_3')$, with $\Delta Z_0 = Z_0 \oplus Z_0'$ and $\Delta W_3 = W_3 \oplus W_3'$
 1  Compute $\Delta X_1$ from $\Delta Z_0$
 2  Compute $\Delta Y_3$ from $\Delta W_3$
 3  for $2^{d^2c/2}$ values for the 🟥 cells in $X_1$ as shown in Figure 4 do
 4      Compute 🟥 cells of $X_1'$ by $X_1' = \Delta X_1 \oplus \Delta X_1$
 5      Compute forward to get the 🟥 cells of $Z_2$ and $Z_2'$
 6      for row $j \in 0, 1, 2, ..., d-1$ in $Z_2$ do
 7          for at row $j$: $2^{dc}$ values of ■ cells in $Z_2$ and $Z_2'$ do
 8              Together with the 🟥 cells in row $j$, compute row $j$ of $Y_3'$, $Y_3$ and
                  $\Delta Y_3$
 9              if the derived row $j$ of $\Delta Y_3$ is not equal to that computed from
                  $\Delta W_3$ in Step 2 then
10                  go to Step 7
11      In this step, all the cells of $Z_2$ and $Z_2'$ are fixed, compute backward to get
          $\Delta X_1$
12      if the ■ cells of $\Delta X_1$ computed from $Z_2$ are equal to that computed from
          $Z_0$ in Step 1 then
13          return the pair as output
```