Triangulating Rebound Attack on AES-like Hashing

Xiaoyang Dong[1,3](✉), Jian Guo[2](✉), Shun Li[2,4](✉), and Phuong Pham[2](✉)

[1] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China
xiaoyangdong@tsinghua.edu.cn
[2] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
guojian@ntu.edu.sg, shun.li@ntu.edu.sg, pham0079@e.ntu.edu.sg
[3] National Financial Cryptography Research Center, Beijing, China
[4] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Abstract. The rebound attack was introduced by Mendel *et al.* at FSE 2009 to fulfill a heavy middle round of a differential path for free, utilizing the degree of freedom from states. The inbound phase was extended to 2 rounds by the Super-Sbox technique invented by Lamberger *et al.* at ASIACRYPT 2009 and Gilbert and Peyrin at FSE 2010. In ASIACRYPT 2010, Sasaki *et al.* further reduced the requirement of memory by introducing the non-full-active Super-Sbox. In this paper, we further develop this line of research by introducing *Super-Inbound*, which is able to connect multiple 1-round or 2-round (non-full-active) Super-Sbox inbound phases by utilizing fully the degrees of freedom from both states and key, yet without the use of large memory. This essentially extends the inbound phase by up to 3 rounds. We applied this technique to find classic or quantum collisions on several `AES`-like hash functions, and improved the attacked round number by 1 to 5 in targets including `AES`-128 and `SKINNY` hashing modes, `Saturnin-Hash`, and `Grøstl-512`. To demonstrate the correctness of our attacks, the semi-free-start collision on 6-round `AES-128-MMO/MP` with estimated time complexity 2^{24} in classical setting was implemented and an example pair was found instantly on a standard PC.

Keywords: Triangulating Rebound · Quantum Computation · Collision Attacks · Rebound Attacks · Triangulation Algorithm · Super-Inbound

1 Introduction

Cryptographic hash function compresses a binary string of arbitrary length into a short fixed-length digest. It is one of the most fundamental cryptographic primitives that act as the underlying building blocks in various advanced cryptographic protocols, including digital signatures, authenticated encryption, secure multiparty computation and post-quantum public-key cryptography, etc.

A cryptographic hash function must fulfill three basic security properties: collision resistance, preimage resistance, and second-preimage resistance. Due to the breakthroughs in collision attacks [76,77,74] on the hash functions MD5 and SHA-1, the academic communities advanced new hashing designs, including the best-known SHA-3 [10]. AES-like hashing, inspired by the elegant yet secure and efficient design strategies of Advanced Encryption Standard (AES) [21], is another new hashing design. It adopts block ciphers or permutations with similar features to AES as the underlying building blocks, such as Whirlpool [6], Grøstl [33], AES-MMO, Grindahl [52], ECHO [8], Haraka v2 [53], etc. These hash functions are commonly known as AES-like hashing.

Rebound Attacks [62], introduced by Mendel, Rechberger, Schläffer, and Thomsen at FSE 2009, is one of the most effective tools on the cryptanalysis of AES-like hash functions. It can be applied to both block cipher based and permutation based constructions. The idea of the rebound attack is to divide an attack into two phases, an inbound and an outbound phase. In the inbound phase, degrees of freedom are used to realize part of the differential characteristic deterministically. The remainder of the characteristic in the outbound phase is fulfilled in a probabilistic manner.

In order to penetrate more rounds, at ASIACRYPT 2009, Lamberger et al. [56] proposed the multiple inbound phases and connected them by leveraging the degrees of freedom of the key. Later, Gilbert and Peyrin [34] and Lamberger et al. [56] further extended the inbound phase by treating two consecutive AES-like rounds as the Super-Sbox [20]. At ASIACRYPT 2010, Sasaki et al. [70] reduced the memory cost by exploiting the differential property of non-full-active Super-Sboxes. At CRYPTO 2011, Naya-Plasencia [66] improved the rebound attack by introducing advanced algorithms for merging large lists. At CRYPTO 2012, Dinur et al. [26] further reduced the memory cost of the rebound attack by proposing the dissection technique. The rebound attack has since become a basic cryptanalysis tool to evaluate hash functions against collision attacks or distinguishing attacks [60,45,71,46,63,25,51,31,59]. Interestingly, the idea of the rebound attack was in turn used to improve the Demirci-Selçuk MITM attacks [24,32] and biclique attack [11].

Quantum Cryptanalysis attracts more and more attention due to Shor's quantum attacks [73] breaking the security of public-key crypto-systems RSA and ECC. For symmetric ciphers, the community has also witnessed many important quantum cryptanalysis results recently, such as quantum distinguisher on 3-round Feistel [54], key-recovery attack on Even-Mansour construction [55], key-recovery attacks or forgeries on MACs and authenticated encryption schemes [47], and more [57,13,27]. However, most attacks need to query the online quantum oracles with superposition states, which is believed to be an impractical projection for quantum physics. Thereafter, the quantum attacks [12,14,48,19,35,67,38] only using offline quantum computers are considered to be of more practical relevance.

Since hash functions can be implemented in offline quantum circuit, the attackers can freely make offline quantum superposition queries. There are several quantum generic (multi-target) preimage attacks [2,19] and multicollision attacks [42,41,58]. For generic quantum collision attacks, three quantum generic algorithms exist under different assumptions of the availability of quantum and classical memory resources:

- Condition 1: Exponentially large quantum random access memory (qRAM) is available. Brassard, Høyer, and Tapp [15] introduced the generic quantum collision attack (named as BHT algorithm) with $2^{n/3}$ quantum time complexity and $2^{n/3}$ qRAM.
- Condition 2: Neither exponentially large qRAM nor classic RAM is available. The quantum version of parallel rho's algorithm [75,9,39] achieves a time-space trade off of time $\frac{2^{n/2}}{S}$ with S computers.
- Condition 3: Exponentially large qRAM is not available but large classical RAM is. Chailloux, Naya-Plasencia, and Schrottenloher [19] introduced the CNS algorithm to find collision in time $2^{2n/5}$ with classical RAM of size $2^{n/5}$.

At EUROCRYPT 2020, Hosoyamada and Sasaki [39] introduced the first quantum dedicated collision attacks by converting the classical rebound attacks into quantum rebound attacks. This showed that, under their respective bounds of generic algorithms, quantum attacks are able to penetrate more rounds than classical attacks. At ASIACRYPT 2020, Dong et al. [29] reduced the requirement of qRAM in the quantum rebound attack by exploiting non-full-active Super-Sbox technique [70]. At CRYPTO 2021, Hosoyamada and Sasaki [40] introduced the quantum collision attacks on reduced SHA-2. At ASIACRYPT 2021, Dong et al. [30] studied the quantum free-start collision attacks.

1.1 Our Contributions

In order to extend the attacked round by the rebound attack, Lamberger et al. [56] connected two local inbound phases by consuming the degree of freedom in the key. Later, Super-Sbox [56,34] and non-full-active Super-Sbox [70] were proposed. In this paper, we further generalize this idea by bridging multiple inbound phases with available degrees of freedom both from the key and the internal data path, to build the *Super-Inbound*. With the help of Khovratovich et al.'s [50] triangulation algorithm, we identify truncated differential trails for `AES` hashing modes with sound configurations on the positions of several local inbound parts, as well as relatively low complexity to bridge the multiple local inbound parts. We name this method as *triangulating rebound attack*. With this advanced method in hand, we achieve the following results:

`AES-MMO`. The MMO-mode (shown in Figure 2) instantiated with `AES` [21] is a popular `AES`-like hashing design, which is standardized by Zigbee [1] and also suggested by ISO [43]. Many advanced cryptographic protocols, e.g., multi-party computation [37,49], adopt `AES-MMO` due to its high efficiency when implemented with `AES-IN`. The best classical collision attacks are for 6 rounds [34,56]. At EUROCRYPT 2020, Hosoyamda and Sasaki [39] introduced two 7-round quantum

collision attacks better than BHT algorithm [15] or parallel rho's algorithm [75]. At ASIACRYPT 2020, Dong et al. [29] introduced 7-round quantum collison better than CNS algorithm [19].

We extend the previous 2-round inbound phase into 3- or 4-round *Super-Inbound* for `AES-128`. Thereby, we build the first 7-round semi-free-start collision attack on `AES-128-MMO/MP` in classical setting, while the previous classical collision attack reaches 6 rounds [34,56]. In addition, a 6-round practical instance of semi-free-start collision and an 8-round quantum semi-free-start collision attack are given. Moreover, the first 8-round quantum collision attack on `AES-128-MMO/MP` is given, with time complexity better than parallel rho's algorithm [75,9,39].

`Saturnin-Hash`. It is the hash function among the post-quantum lightweight cipher family `Saturnin` designed by Canteaut et al. [17]. It is one of the round 2 candidates of the NIST lightweight cryptography competition. The designers of `Saturnin` introduced the 10-round related-key recovery attack on the `Saturnin` block cipher [16]. At ASIACRYPT 2021, Dong et al. [30] proposed several quantum and classical collision attacks on `Saturnin-Hash` and its compression function, including a 7-round semi-free-start quantum collision attack and an 8-round free-start quantum collision attack.

We extend the previous 2-round inbound phase to the 4-round *Super-Inbound*. Finally, the time complexity of the 8-round free-start quantum collision attack is significantly reduce from $2^{122.5}$ to $2^{89.65}$. Further, the first 10-round free-start quantum collision attack is derived with two more rounds than Dong et al.'s result [30].

`SKINNY-128-384-MMO`. As `SKINNY` [7] is running for ISO standard, it is natural to build lightweight hash with the ISO suggested hashing mode MMO. At ASIACRYPT 2021, Dong et al. [30] introduced the 16-round quantum free-start collision attack on `SKINNY-128-384-MMO`. In this paper, by exploring the large degrees of freedom of the key schedule for `SKINNY-128-384-MMO`, we extend the previous 2-round inbound phase into a 5-round *Super-Inbound*, and finally achieve 5 more rounds on the free-start quantum collision attack. In addition, a 19-round classical free start collision attack is given, which is even better than previous quantum attack.

`Grøstl`. `Grøstl` is one of the five finalists of SHA-3. In this paper, by bridging the differences with degrees of freedom from the states, we propose the memoryless method to solve the 3-round non-full-active inbound part for `Grøstl-512`. Thereafter, the improved semi-free-start collisions with one more round than before for both `Grøstl-512` and its predecessor are derived. The results are summarized in Table 1.

1.2 Novelty and Comparison with Previous Works

Both rebound attack [62] and triangulation algorithm [50] are existing techniques. However, combining these two techniques together as one cryptanalysis

tool has not appeared before. When using these two techniques together, we can exploit many more degrees of freedom (DoF) from the key schedule algorithm, which greatly extends the inbound part of our rebound attack. Therefore, the number of possible truncated differential trails for the rebound attack increases significantly than before. In previous rebound attacks on AES-MMO (including [39,29]), the authors place a 2-round full/non-full active Super-Sbox in the inbound phase, and extend it forwards and backwards in the outbound phases to find a useful rebound-attack trail. However, in our attack, we have to use 3-/4-round truncated differentials as the multiple inbound phases, which have many more choices than a 2-round Super-Sbox before. Moreover, not all those 3-/4-round inbound can be solved efficiently by triangulation algorithm (actually, it is time consuming to compute a compatible pair for most 3-/4-round truncated differentials), and we have to pick a good trail from many choices.

Therefore, as another contribution, we introduce a heuristic automatic tool (CP-based) to identify good trails for our rebound attacks, which succeeds to find the 7-/8-round collision attacks on AES-MMO and gains 1-round improvement.

We note that the multiple inbound phases technique was already introduced and used many times before, see for instance the rebound attacks on the LANE [59], ECHO [44], JH [68], etc. The additional degrees of freedom were in these cases given by bigger states. The truncated differences affect both the number of rounds covered by the inbound phases, and the differential probabilities of the outbound phases. In the previous works, the choices of these truncated differences are made in ad hoc ways. In our paper, we introduce triangulation algorithm into the rebound attack, as well as the automated search of configuration for the rebound attack with multiple inbound phases, which is the main novelty comparing to previous ad hoc ways.

2 Preliminaries

2.1 AES-like Hashing

AES [21] operates on a 4×4 column-major order array of bytes, whose round function contains four major transformations as illustrated in Figure 1: SubBytes (SB), ShiftRows (SR), MixColumns (MC), and AddRoundKey (AK). By making changes to the numbers of rows and columns, the sizes of the cells, the order of the transformations, one can produce new designs named as AES-like round functions. Usually, the MixColumns is to multiply an MDS matrix to each column of the state. An exception is SKINNY [7], which uses the non-MDS matrix.

By using (keyed) permutations with AES-like round functions in certain hashing modes (like MD, MMO, and MP hashing modes [64, Section 9.4] in Figure 2), compression functions (denoted as CF) can be constructed. Plugging such compression functions into the Merkle-Damgård construction [65,22], one arrives at AES-like hashings. Concrete designs include AES-128-MMO, AES-128-MP, Saturnin-Hash [17], Whirlpool [6] Grøstl [33].

Table 1: A summary of the results.

Target	Attack	Rounds	Time	C-Mem	qRAM	Setting	Ref.
AES-128-MMO/MP							
Hash	Collision	5/10	2^{56}	2^{4}	0	Classic	[62]
		6/10	2^{56}	2^{32}	0	Classic	[34,56]
		7/10	$2^{42.50}$	0	2^{48}	Quantum	[39]
		7/10	$2^{59.5}$	0	0	Quantum	[39]
		7/10	$2^{45.8}$	0	0	Quantum	[29]
		8/10	$2^{55.53}$	0	0	Quantum	Sect. 4.3
	Preimage	7/10	2^{125}	2^{8}	-	Classic	[69]
		7/10	2^{123}	2^{8}	-		[3]
		8/10	2^{125}	2^{8}	-		[4]
Compression function	Semi-free	6/10	2^{24}	-	-	Classic	Sect. 4.2
	Semi-free	7/10	2^{56}	2^{16}	-	Classic	Sect. 4.1
	Semi-free	8/10	2^{34}	-	-	Quantum	Sect. 4.2
	any	any	2^{64}	-	-	any	[75,39,9]
	any	any	$2^{42.7}$	-	$2^{42.7}$	Quantum	[15]
	any	any	$2^{51.2}$	$2^{25.6}$	-	Quantum	[19]
Saturnin-Hash							
Hash	Collision	5/16	2^{64}	2^{66}	0	Classic	[30]
		7/16	$2^{113.5}$	-	-	Quantum	[30]
	Preimage	7/16	2^{232}	2^{48}	-	Classic	[28]
Compression function	Free-start	6/16	2^{80}	2^{66}	-	Classic	[30]
	Semi-free	7/16	$2^{90.1}$	-	-	Quantum	[30]
	Semi-free	7/16	2^{86}	-	-	Quantum	Sect. D
	Free-start	8/16	$2^{122.5}$	-	-	Quantum	[30]
	Free-start	8/16	$2^{89.65}$	-	-	Quantum	Sect. 5.1
	Free-start	10/16	$2^{127.2}$	-	-	Quantum	Sect. 5.2
	any	any	2^{128}	-	-	any	[75,39,9]
	any	any	$2^{85.3}$	-	$2^{85.3}$	Quantum	[15]
	any	any	$2^{102.4}$	$2^{51.2}$	-	Quantum	[19]
Grøstl-512							
Compression function	Distinguisher	10/14	2^{392}	2^{64}	-	Classic	[45]
		11/14	2^{72}	2^{56}	-	Classic	[18]
	Semi-free	6/14	2^{180}	2^{64}	-	Classic	[72]
		7/14	2^{214}	-	-	Quantum	Sect. G
Compression function v0	Semi-free	7/14	2^{152}	2^{64}	-	Classic	[61]
		7/14	2^{152}	2^{56}	-	Classic	[70]
		8/14	2^{244}	-	-	Quantum	Sect. G
	any	any	2^{256}	-	-	any	[75,39,9]
	any	any	$2^{170.7}$	-	$2^{170.7}$	Quantum	[15]
	any	any	$2^{204.8}$	$2^{102.4}$	-	Quantum	[19]
SKINNY-128-384-MMO/MP							
Compression function	Free-start	16/56	$2^{59.8}$	-		Quantum	[30]
		19/56	$2^{51.2}$	-		Classic	Sect. 6.2
		21/56	$2^{46.2}$	-		Quantum	Sect. 6.1
	any	any	2^{64}	-		any	[75,39,9]
	any	any	$2^{42.7}$	-	$2^{42.7}$	Quantum	[15]
	any	any	$2^{51.2}$	$2^{25.6}$	-	Quantum	[19]

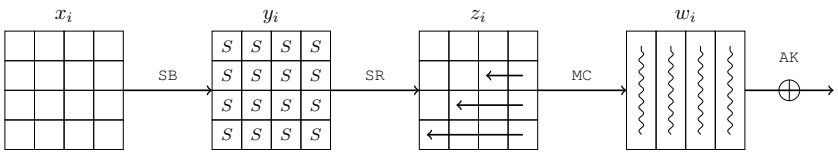
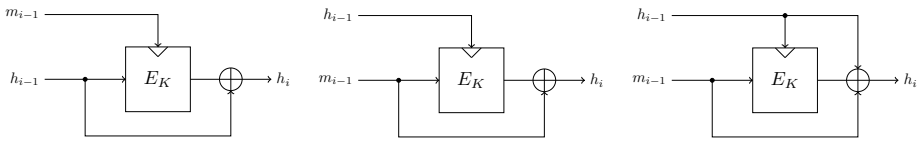


Fig. 1: The round function of AES



(DM) Davies-Meyer (MMO) Matyas-Meyer-Oseas (MP) Miyaguchi-Preneel

Fig. 2: Common Hashing Modes

2.2 The Rebound Attack

The rebound attack was first introduced by Mendel et al. in [62]. It consists of an inbound phase and an outbound phase as shown in Figure 3, where F is an internal block cipher or permutation which is split into three subparts, then $F = F_{fw} \circ F_{in} \circ F_{bw}$.

- **Inbound phase**. In the inbound phase, the attackers efficiently fulfill the low probability part in the middle of the differential trail with a meet-in-the-middle technique. The degree of freedom is the number of matched pairs in the inbound phase, which will act as the starting point for the outbound phase.
- **Outbound phase**. In the outbound phase, the matched values of the inbound phase, i.e., starting points, are computed backward and forward through F_{bw} and F_{fw} to obtain a pair of values which satisfy the outbound differential trail in a bruteforce fashion.

Overall, the rebound attack is essentially a technique to efficiently generate a message pair fulfilling the inbound phase, which utilizes a truncated differential rather than a single differential characteristic. Suppose the probability of the outbound phase is p, then we have to prepare $1/p$ starting points in the inbound phase to expect one pair conforming to the differential trail of the outbound phase. Hence, the degree of freedom should be larger than $1/p$.

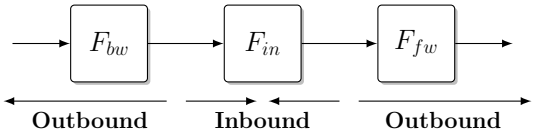


Fig. 3: The Rebound Attack

2.3 The Super-Sbox Technique

The Super-Sbox technique proposed by Gilbert and Peyrin [34] and Lamberger et al. [56] extends Mendel et al.'s [62] inbound part into 2 S-box layers by regarding them as four Super-Sboxes as shown in Figure 4 (a). In [70], Sasaki et al. further reduced the the memory complexity by considering non-full-active Super-Sboxes as shown in Figure 4 (b). In both the two techniques, AK acts as a constant addition operation and does not provide any degree of freedom.

(a) Super-Sbox (b) Non-full-active Super-Sbox

Fig. 4: The Two-Round Differential

Super-Sbox Technique. For the jth Super-Sbox $\mathtt{SSB}_j$ and given input difference $\Delta x_i^{(j)}$ ($j = 0$ in Figure 4 (a)), compute $\Delta y_{i+1}^{(j)} = \mathtt{SSB}_j(x \oplus \Delta x_i^{(j)}) \oplus \mathtt{SSB}_j(x)$ for $x \in \mathbb{F}^{32}$. Store the pair $(x, x \oplus \Delta x_i^{(j)})$ in a table $\mathbb{L}^{(j)}[\Delta y_{i+1}^{(j)}]$. Given $\Delta y_{i+1}^{(j)}$, we find a pair conforming the two-round differential with $(\Delta x_i^{(j)}, \Delta y_{i+1}^{(j)})$ by assessing $\mathbb{L}^{(j)}[\Delta y_{i+1}^{(j)}]$. The memory cost is about 2^{32}.

Non-full-active Super-Sbox. In Figure 4 (b), the Property 1 of MDS in MC is used. Look at $\Delta w_i = \mathtt{MC}(\Delta z_i)$, by guessing the differences of one active byte, we can determine other differences according to Property 1. Then, for a fixed input-output differences $(\Delta x_i^{(j)}, \Delta y_{i+1}^{(j)})$ of $\mathtt{SSB}_j$, we deduce all the input-output differences for the active cells of two S-box layers for each guess and then deduce their values by accessing the differential distribution table (DDT) of the S-box.

Property 1. Suppose MC is an $n \times n$ MDS matrix and $\mathtt{MC} \cdot (z[1], \ldots, z[n])^T = (w[1], \ldots, w[n])^T$, the knowledge of any n out of $2n$ bytes of (z, w) is necessary and sufficient to determine the rest. (z, w) here can be either value or difference.

2.4 Triangulation Algorithm

At CT-RSA 2009, Khovratovich, Biryukov, and Nikolić [50] introduced the triangulation algorithm tool, an efficient Gaussian-based-algorithm to solve system of bijective equations, to automatically detect a way to solve the nonlinear system. The algorithm models an AES-like block cipher as a system of key schedule and round function equations, and the state bytes and key bytes as variables. At first, the system is determined by n initial state bytes and k initial key bytes. Therefore, when m bytes are fixed, the algorithm will return $n + k - m$ "free variables" to form a basis of the system. The algorithm can output exactly the "free variables" when the system is deterministically solvable, and improve the

guessing and checking the "free variables" when the system is probabilistically solvable. The idea of triangulation algorithm is roughly described following.

1. Construct a system of equations whose variables are the bytes. The predefined values are fixed to constants.
2. All variables and equations are marked as non-processed.
3. Mark the variable which is involved in only one non-processed equation as processed. Also mark this equation as processed. If no such variable exist, exit.
4. Return to Step 3 if still have non-processed equations.
5. Return all non-processed variables as "free variables".

After the "free variables" are identified, we randomly assign values for them and deduce a solution for the whole nonlinear system. For details, please refer to [50].

2.5 Collision Attacks and Its Variants

Given a hash function H, a standard collision message pair (m, m') satisfies $H(IV, m) = H(IV, m')$, where the initial vector IV is a fixed inital value. A *semi-free-start collision* is to find a pair (u, m) and (u, m'), such that $H(u, m) = H(u, m')$ $(u \neq IV)$. A *free-start collision* is to find a pair (v, m) and (v', m'), so that $H(v, m) = H(v', m')$ $(v \neq v')$. When the hash function H is built by iterating the compression function (`CF`) with Merkle-Damgård construction, we can similarly define the semi-free-start collision and free-start collision attack on the compression function. Taking the `MMO` and `MP` hashing modes in Figure 2 as examples, when considering the semi-free-start or free-start collision attack, the attackers can explore the degrees of freedom from the chaining value h_{i-1} through the key shedule algorithm, which may lead to better attacks such as [56,71]. We would like to emphasize the importance of semi-free-start and 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 semi-free-start and free-start collisions.

3 Triangulating Rebound Attack

3.1 Solving Non-Full-Active Super-Sbox by Key Bytes

In previous collision attacks, both Super-Sbox [56,34] and non-full-active Super-Sbox [70] techniques, the subkeys are prefixed in advance (as shown in Section 2.3). In other words, the degree of freedom of the subkeys was not be utilized. We observe that, by carefully choosing the values of subkeys, state pairs conforming the given differential can be obtained without additional computation only at a cost of memory for storing the differential distribution table (`DDT`). This observation enables utilization of the degrees of freedom from the corresponding subkeys and leads to collision attacks without the need of building large lists of compatible pairs for Super-Sboxes. Here we describe this new Super-Sbox technique through a concrete example in Figure 5.

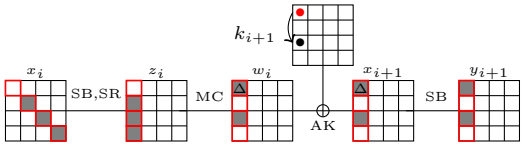


Fig. 5: An example of Super-Sbox with degrees of freedom from the subkey

Given the pair of non-full difference of $(\Delta x_i, \Delta y_{i+1})$ and zero difference in key of an Super-Sbox (marked as red for the corresponding column), we are to generate state pairs and corresponding subkey k_{i+1} conforming to that difference. First, we randomly assign a difference Δ compatible with $\Delta y_{i+1}[0]$ to $\Delta x_{i+1}[0]$ and asset to `DDT` to get an actual value for $x_{i+1}[0]$ conforming the difference $(\Delta, \Delta y_{i+1}[0])$. Since there is no difference in the subkey k_{i+1}, the difference $\Delta w_i[0]$ is equal to Δ as well. We use a simple property of maximum distance separable (MDS) matrix to determine all the rest bytes of Δw_i and Δz_i.

With the knowledge of 4 bytes of Δz_i and Δw_i ($\Delta z_i[0] = \Delta w_i[1,3] = 0$ and $\Delta w_i[0] = \Delta$), all bytes of Δz_i and Δw_i are uniquely determined, and so for the entire differential characteristic $(\Delta x_i, \Delta z_i, \Delta w_i, \Delta x_{i+1}, \Delta y_{i+1})$ of the Super-Sbox. By looking up `DDT`, the byte values before and after the active Sboxes, e.g., $x_i[5, 10, 15], z_i[1, 2, 3], x_{i+1}[0, 2]$ and $y_{i+1}[0, 2]$, are determined as well. Next, we randomly assign a value to $k_{i+1}[0]$, then $w_i[0]$ can be calculated as $w_i[0] = x_{i+1}[0] \oplus k_{i+1}[0]$. Again, Property 1 allows to determine the remaining bytes $z_i[0]$ and $w_i[1, 2, 3]$ since 4 bytes $z_i[1, 2, 3]$ and $w_i[0]$ are known. Finally, the key byte $k_{i+1}[2]$ is determined as $k_{i+1}[2] = w_i[2] \oplus x_{i+1}[2]$. In summary, after randomly assigning (compatible) values to $\Delta x_{i+1}[0]$ and $k_{i+1}[0]$, all the values and differences of the active bytes in the state as well as the subkey bytes corresponding to the active state bytes positions of `AK` are determined, with time complexity 1.

3.2 Connecting Multiple Inbound Phases by Key Bytes

In this section, we present a 3-round inbound phase by extending the Super-Sbox backward by one round. Since the Super-Sbox and the extended round are not solved in two inbound phases, this technique is also referred to as "multiple inbound phases". These phases are connected by free bytes of the corresponding subkeys, and so one has to ensure the value assignments to the subkeys from different rounds are not over-defined through the key schedule algorithm. This new technique is illustrated through an example of 3-round truncated differential trail depicted in Figure 6.

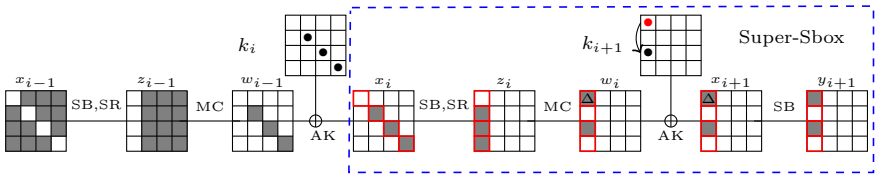


Fig. 6: An example of 3-round multiple inbound phases

Given the differences Δx_{i-1}, Δx_i, Δy_{i+1} and zero difference in key, we are to generate state and key pairs conforming the differential path with low memory requirement by utilizing the degrees of freedom from subkeys k_i and k_{i+1}. First, applying the Super-Sbox technique presented above to the given Δx_i and Δy_{i+1}, both value and difference of $\Delta x_i[5, 10, 15]$ and value of $k_{i+1}[0, 2]$ will be determined. Then, with fixed Δx_{i-1} and Δx_i, standard inbound phase of the original rebound attack can be applied, i.e., lookup DDT with input/output differences $(\Delta x_{i-1}, \mathtt{SR}^{-1}(\Delta z_{i-1} = \mathtt{MC}^{-1}(\Delta x_i)))$, values of the last three columns of z_{i-1} and so w_{i-1} are determined. Hence, $\Delta k_i[5, 10, 15]$ can be calculated as $\Delta w_{i-1}[5, 10, 15] \oplus \Delta x_i[5, 10, 15]$. Note, there is no direct implication between any assignment of $k_i[5, 10, 15]$ and $k_{i+1}[0, 2]$ through the AES key schedule. The whole process costs 17 DDT lookups and memory hosting the DDT lookup table.

3.3 The Triangulating Rebound Attack

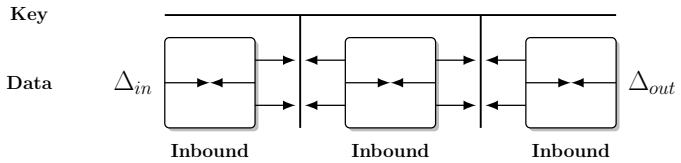


Fig. 7: *Super-Inbound*: bridging multiple local inbound parts

Generally speaking, the development of the rebound attack can be viewed as a continuous process to find as many degrees of freedom (DoF) as possible, to either reduce the overall inbound solving complexity, or to extend for more rounds. As shown in Figure 7, the multiple local inbound parts form a *Super-Inbound*. Given the input/output differences $(\Delta_{in}, \Delta_{out})$ of the super inbound phase, bridging the multiple local inbound parts is to find the conforming pair for $(\Delta_{in}, \Delta_{out})$, while the exact differentials and values of the internal rounds are up to attacker's control. Usually, $(\Delta_{in}, \Delta_{out})$ are chosen from a truncated differential in the rebound attack. In summary, the source of DoF for the whole rebound attack can come from the choices of

S1 truncated differential of the internal rounds of the *Super-Inbound*,
S2 input/output differences and values of the active Sboxes of the state,
S3 values of the inactive Sboxes of the state,
S4 differences of key bytes,
S5 values of key bytes.

The differences and values of the active Sboxes of the state are listed together, as once differences are fixed, values can be derived from DDT, and vice versa. This is not the case for key bytes, as it is not necessarily for each of the key bytes to

pass through Sbox depending on the key schedule algorithm. Then the Super-Sbox technique can be viewed as a way to connect two 1-round inbound parts by fully utilizing the DoF from S2, i.e., the output differences of first round and input differences of the second round. Our techniques presented in Section 3.1, 3.2 is from (S2,S5), (S2,S5), respectively. Under this generic view, connecting multiple inbound parts to form a *Super-Inbound* can be seen as a system of nonlinear equations, where all DoF serve as variables and DoF from subkeys are constrained by key schedule, and our goal is to find a procedure to solve this system, efficiently. There are two main considerations.

I. The system of nonlinear equations must have solutions. In other words, the system should not be over-defined. Hence, the number DoF should be larger than the number of equations.
II. The system must be solvable efficiently, which for most of the time can be translated into a procedure of consuming degrees of freedom by fixing certain variables involved, and the final complexity is determined by the number variables which can not be fixed and have to be brute-forced.

We adopt the triangulation algorithm [50] as shown in Section 2.4 to solve the nonlinear system and bridge the multiple inbound parts. The algorithm can output exactly the "free variables" when the system is deterministically solvable, and improve the guessing and checking the "free variables" when the system is probabilistically solvable. We name this method as *triangulating rebound attack*.

Heuristic Method to Determine a Trail. So far, we have not mentioned the use of DoF from S1 and S4 yet. While S4 determines whether difference is allowed in key hence resulting in semi-free-start or free-start collision for attacks against hash functions, S1 directly determines the system of nonlinear equations, hence the number of rounds covered by the *Super-Inbound* and complexity as well. For collision attack on n-bit hash, the overall steps to identify a rebound attack trail are as follows:

1. Find a truncated differential trail following the *Constrained Programming* based search model from [5].
2. Choose certain consecutive rounds as the *Super-Inbound* (i.e., S1), which includes several local inbound parts. For example, in Figure 7 a *Super-Inbound* includes three local inbound parts. Usually, 2 to 4 consecutive rounds are chosen as possible *Super-Inbound*. Following this search, all the input/output differences of the active Sboxes are fixed, and therefore corresponding values are fixed.
3. Build equation system that connects the inbound parts and through key schedule. Check if the system is solvable by triangulation algorithm.

Before checking the solvability of the system, a potential truncated trail must satisfy the condition that the number of active Sboxes in the *Super-Inbound* does not exceed the DoF from both key and state. This is because the system of state

bytes and keys bytes are determined by the initial k key bytes and n plaintext bytes, then for any set of fixed bytes in advance, the system is potentially solvable if the number of fixed bytes is not larger than $n+k$. Our observation shows that the ideal case is when the number of active bytes in the *Super-Inbound* equals to $n+k-1$ and only 1 free byte is left, since the scenario without free byte is likely to form an over-defined system. Therefore, this constraint is also added to our model. For classical collision attacks, after the *Super-Inbound* is fixed, the probability of the outbound phase can be computed and denoted as Pr, which should meet $Pr > 2^{-n/2}$. For quantum collision attacks, let Pr[5] $> 2^{-n}$. Similarly to previous models [39], for the attacks based on the truncated differentials, P_r is the probability for the cancellation of `MC` operator in the backward and forward trunks, as well as the cancellation for the feed-forward operator of hashing modes. After checking the potential truncated trails satisfying the above conditions for inbound and outbound phases, the triangulation algorithm is applied to detect the free bytes left followed by the step to generate sufficient data pairs for the outbound phase.

4 Improved Collision Attacks on `AES-128-MMO`

In this section, by applying the *triangulating rebound*, we introduce a 7-round classical semi-free-start collision attack and an 8-round quantum semi-free-start collision attack on `AES-MMO`. Moreover, we identify the first 8-round quantum collision attack, which is better than parallel rho's algorithm [75].

4.1 Semi-Free-Start Collision Attack on 7-Round `AES-128`

New 7-round `AES-128` Truncated Differential Trail. We introduce here our new differential trail in Figure 8. Our Super-Inbound phase covers 3 middle rounds with 2 inbound phases (marked with red and blue dashed lines), which consumes 31 bytes of degree of freedom for the fixed state bytes, i.e., 16 active bytes in x_3, 9 active bytes in x_4, and 6 active bytes in x_5. Since we have 32 bytes in total (16 bytes in key and 16 bytes in the state), only 1 free byte is left. By triangulating algorithm, the 1 free byte is identified to be $k_5[12]$. The remaining outbound phase happens with probability $p_{out} = 2^{-56}$ including the `MC` cancellation in round 1 and 4-byte cancellation for $\Delta P = \Delta C$.

The Super-Inbound Phase. As shown in Figure 8, the Super-Inbound phase divides into 2 parts, the Inbound II marked with blue dashed line makes use of k_5 to link round 4 and round 5, and the Inbound I marked with red dashed line connects with Inbound II via k_4. To generate a data pair conforming a given difference $(\Delta z_2, \Delta w_3, \Delta w_5)$, assuming $\Delta Z_5 = \texttt{MC}^{-1}(\Delta w_5)$ keeps the truncated differential in Figure 8, we perform the following steps:

[5] As shown in the introduction, we consider three quantum attack conditions. $Pr > 2^{-2n/3}$ is to be better than the BHT algorithm, $Pr > 2^{-n}$ is to be better than quantum time-space tradeoff, $Pr > 2^{-4n/5}$ is to be better than CNS algorithm. So we let $Pr > 2^{-n}$ to keep all characteristics that may lead to possible attacks.

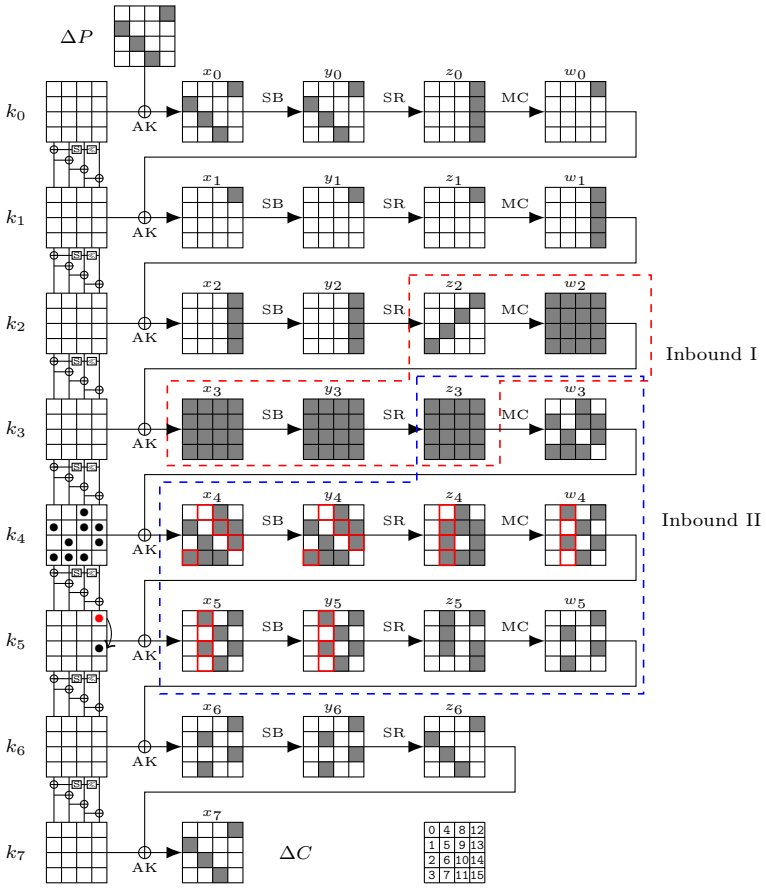


Fig. 8: Semi-free-start collision attack on 7-round AES-128

– Perform the Inbound II:
  1. Randomly assign a difference to $\Delta x_5[4, 9, 12]$. Then $\Delta w_4[4, 9, 12]$ are known accordingly, implies all the differences at these active cells of z_4 and the remaining active cells of w_4 are determined as a result of the Property 1. From Δz_4, we deduce $\Delta y_4 = \mathtt{SR}^{-1}(\Delta z_4)$.
  2. Compute the difference $\Delta y_5 = \mathtt{SR}^{-1}(\mathtt{MC}^{-1}(\Delta w_5))$. Next, we deduce the values of the active cells at x_4 and x_5 by assessing to the DDT for each cell of $(\Delta x_4, \Delta y_4)$ and $(\Delta x_5, \Delta y_5)$. The active byte values of y_4 and y_5 are determined respectively.

– Perform the Inbound I:
  1. Compute $\Delta x_3 = \mathtt{MC}(\Delta z_2)$ and $\Delta y_3 = \mathtt{SR}^{-1}(\mathtt{MC}^{-1}(\Delta w_3))$.
  2. Deduce full state values x_3 and y_3 by assessing the DDT. We compute the full state values of w_3 by the action $w_3 = \mathtt{MC}(\mathtt{SR}(y_3))$.

– Connect the phases by the subkeys:

1. Since the full state of w_3 and the active bytes of x_4 are known, we obtain $k_4[1,3,6,7,8,9,11,13,14] = x_4[1,3,6,7,8,9,11,13,14] \oplus w_3[1,3,6,7,8,9,11,13,14]$ (marked as black dots).
2. Add constrains on the related cells of k_5 to the system of equations of the subkeys. We solve the constrains on k_4 and k_5 by triangulation algorithm and obtain a free byte $k_5[12]$.
3. The steps (shown in Figure 9) to recover k_5 and compute a starting point are shown as follows (equations are also given in Table 2) :
   (a) As shown in Figure 9(1), all the differences marked by 'D' are known. In Figure 9(2), by assessing the DDT, we know the values of $x_3[V]$, $x_4[V]$, and $x_5[V]$. Then deduce $k_4[V] = w_3[V] \oplus x_4[V]$.
   (b) In Figure 9(3), randomly assign a value for $k_5[12]$ marked by 'V', compute $w_4[12] = k_5[12] \oplus x_5[12]$. According to Property 1, deduce all the bytes marked by 'V' in z_4 and w_4. Then, $k_5[14] = x_5[14] \oplus w_4[14]$. From $z_4[V]$, we get $x_4[V]$. Then $k_4[V]$ is deduced.
   (c) In Figure 9(4), following the key schedule of AES-128, compute the $k_5[V]$: $k_5[8] = k_4[12] \oplus k_5[12]$, $k_5[10] = k_4[14] \oplus k_5[14]$, $k_5[11] = k_4[11] \oplus k_5[7]$, $k_5[4] = k_4[8] \oplus k_5[8]$, $k_5[7] = k_5[3] \oplus k_4[7]$, $k_5[1] = k_4[1] \oplus \mathtt{SB}(k_4[14]) \oplus r_c$, $k_5[3] = k_4[3] \oplus \mathtt{SB}(k_4[12]) \oplus r_c$.
   (d) In Figure 9(5), deduce $w_4[4] = k_5[4] \oplus x_5[4]$ and $w_4[11] = k_5[11] \oplus x_5[11]$, then acorrding to Property 1, all the bytes in w_4 and z_4 marked by 'V' are deduced. Then, two bytes marked by 'V' in k_5 are deduced, i.e., $k_5[6] = w_4[6] \oplus x_5[6]$ and $k_5[9] = w_4[9] \oplus x_5[9]$. Moreover, compute two bytes marked by 'V' in k_4, i.e., $k_4[2] = x_4[2] \oplus w_3[2]$ and $k_4[4] = x_4[4] \oplus w_3[4]$.
   (e) In Figure 9(6), following the key schedule, deduce the bytes marked by 'V' in k_5, i.e., $k_5[0] = k_4[4] \oplus k_5[4]$, $k_5[2] = k_4[6] \oplus k_5[6]$, $k_5[5] = k_4[9] \oplus k_5[9]$, $k_5[13] = k_4[13] \oplus k_5[9]$, $k_4[15] = \mathtt{SB}^{-1}(k_4[2] \oplus k_5[2])$, $k_5[15] = k_4[15] \oplus k_5[11]$.
4. Finally, full state w_3 and full subkey k_5 are obtained by using 34 DDT assesses.

Due to the probability of the outbound phase is 2^{-56}, we need to check 2^{56} starting points to find the collision. The final complexity is 2^{56} and memory needed is 2^{16} for DDT storing.

Remark. In a rebound attack [62], we typically have a probability $\frac{1}{2}$ to have a solution for each active SBox, so that we have to repeat the attack with new differences, but when the differences are compatible we obtain many solutions and the amortized cost is close to 1. In our attack, when $(\Delta x_3, \Delta y_3)$, $(\Delta x_4, \Delta y_4)$ and $(\Delta x_5, \Delta y_5)$ are fixed, we have 31 active Sboxes. Therefore, one compatible differential characteristic is found in roughly 2^{31} time, which leads to 2^{31} solutions for the active bytes of (x_3, x_4, x_5). Moreover, since we can randomly assign $k_5[12]$ (Step 3 (b)), we totally have about $2^{31+8} = 2^{39}$ starting points, which is lower than the final time 2^{56}. Briefly speaking, for a given $(\Delta x_3, \Delta y_3)$, $(\Delta x_4, \Delta y_4)$, $(\Delta x_5, \Delta y_5)$ and $k_5[12]$, we find one starting point with time complexity of 1 on average.

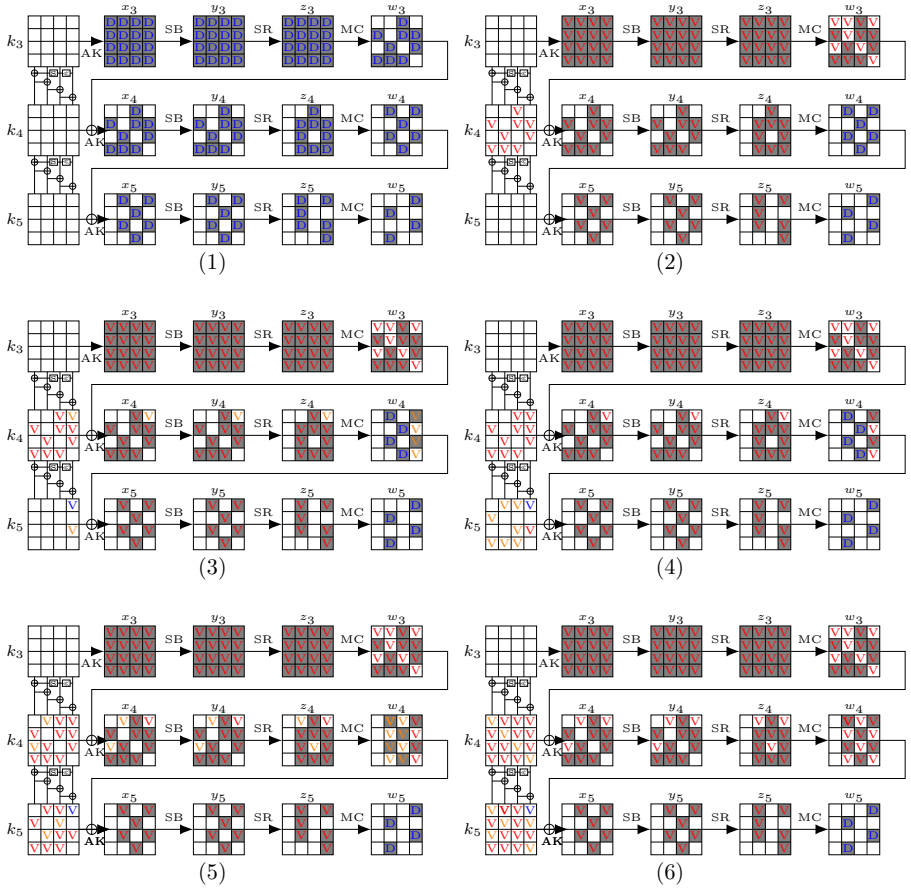


Fig. 9: Steps to recover k_5. The bytes marked by 'D' mean that the differences are known. The bytes marked by 'V' mean that the values are known. In subfigures, the bytes marked by 'V' are deduced from those marked by 'V'.

1. $w_4[12] = x_5[12] \oplus k_5[12]$	15. $k_5[7] = k_5[3] \oplus k_4[7]$
2. $z_4[12], w_4[13, 14, 15] = \texttt{MC}(w_4[12], z_4[13, 14, 15])$	16. $k_5[11] = k_4[11] \oplus k_5[7]$
3. $k_5[14] = w_4[14] \oplus x_5[14]$	17. $w_4[11] = k_5[11] \oplus x_5[11]$
4. $x_4[12] = \texttt{SB}^{-1}(z_4[12])$	18. $z_4[10], w_4[8, 9, 10] = \texttt{MC}(z_4[8, 9, 11], w_4[11])$
5. $k_4[12] = w_3[12] \oplus x_4[12]$	19. $k_5[9] = w_4[9] \oplus x_5[9]$
6. $k_5[8] = k_4[12] \oplus k_5[12]$	20. $k_5[13] = k_5[9] \oplus k_4[13]$
7. $k_5[1] = k_4[1] \oplus \texttt{SB}(k_4[14]) \oplus r_c$	21. $k_5[5] = k_4[9] \oplus k_5[9]$
8. $k_5[10] = k_4[14] \oplus k_5[14]$	22. $k_4[5] = k_5[1] \oplus k_5[5]$
9. $k_5[4] = k_4[8] \oplus k_5[8]$	23. $k_4[4] = w_3[4] \oplus \texttt{SB}^{-1}(z_4[4])$
10. $w_4[4] = k_5[4] \oplus x_5[4]$	24. $k_5[0] = k_4[4] \oplus k_5[4]$
11. $z_4[4], w_4[5, 6, 7] = \texttt{MC}^{-1}(w_4[4], z_4[5, 6, 7])$	25. $x_4[2] = \texttt{SB}^{-1}(z_4[10])$
12. $k_5[6] = w_4[6] \oplus x_5[6]$	26. $k_4[2] = w_3[2] \oplus x_4[2]$
13. $k_5[2] = k_4[6] \oplus k_5[6]$	27. $k_4[15] = \texttt{SB}^{-1}(k_4[2] \oplus k_5[2])$
14. $k_5[3] = k_4[3] \oplus \texttt{SB}(k_4[12]) \oplus r_c$	28. $k_5[15] = k_5[11] \oplus k_4[15]$

Table 2: Steps to recover the subkey k_5 from known bytes

4.2 Semi-Free-Start Collision Attack on 8-Round AES-128-MMO

As shown in Figure 10, there are three inbound parts in the Super-Inbound phase. For given fixed differences of Δz_1, Δz_2, Δw_4, and Δw_5, assuming $\Delta w_1 = \texttt{MC}(\Delta z_1)$ and $\Delta z_5 = \texttt{MC}^{-1}(\Delta w_5)$ keep the truncated differential of Figure 10, we perform inbound part I, II, and III without the key information. Then, we connect the three inbound part by determining certain key values and get the starting point for the outbound phase. The probability of the outbound phase is 2^{-64}, including the condition $\Delta P = \Delta C$.

Given fixed differences of Δz_1, Δz_2, Δw_4, and Δw_5, the procedures to get one starting point are as follows:

1. Deduce values $(x_5[0, 5, 10], x'_5[0, 5, 10])$ with Δw_4 and $\Delta z_5 = \texttt{MC}^{-1}(\Delta w_5)$ by accessing the DDT.
2. Deduce values $(x_2[0, 1, 2], x'_2[0, 1, 2])$ with $\Delta w_1 = \texttt{MC}(\Delta z_1)$ and Δz_2 by accessing the DDT. Then $z_2[0, 10, 13]$ are known as well.
3. For $\texttt{SSB}^{(0)}$ in Inbound III (marked in red), we compute $\Delta x_3 = \texttt{MC}(\Delta z_2)$ and $\Delta y_4 = \texttt{SR}^{-1}(\texttt{MC}^{-1}(\Delta w_4))$.
4. Randomly choose $\Delta w_3[0, 2] \in \mathbb{F}_2^8 \times \mathbb{F}_2^8$,
   (a) Compute other active differences $\Delta z_3[0, 2, 3]$ and $\Delta w_3[3]$ by Property 1
   (b) Deduce $(z_3[0, 2, 3], z'_3[0, 2, 3])$ and $(x_4[0, 2, 3], x'_4[0, 2, 3])$ by assessing DDT
5. Repeat Step 4 for random differences $\Delta w_3[4, 5], \Delta w_3[8, 9], \Delta w_3[13, 14]$ to compute all active bytes of x_3 and x_4. Next, we compute $w_4[0, 5, 10] = \texttt{MC}(\texttt{SR}(\texttt{SB}(x_4)))[0, 5, 10]$ and then $k_5[0, 5, 10] = w_4[0, 5, 10] \oplus x_5[0, 5, 10]$.
6. Assign random values to $k_3[6]$, $k_4[0]$, and $k_4[4]$. We recover the full key k_5 by the following steps in Table 3. There exists a filter of 2^{-8} in the step to recover $k_3[3]$.

Since Step 6 has a filter of 2^{-8} to meet the condition "$k_3[3] = w_2[3] \oplus x_3[3] \stackrel{?}{=} \texttt{SB}(k_3[12]) \oplus k_4[3] \oplus r_c$ " in Table 3, the time complexity to find a starting point

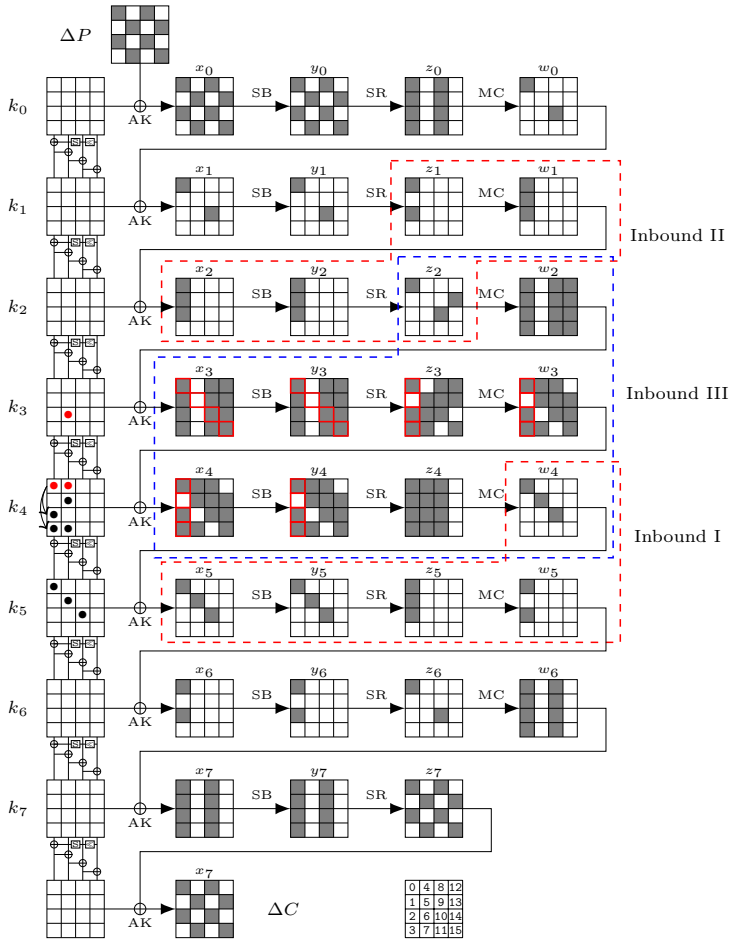


Fig. 10: Quantum semi-free-start collision attack on 8-round `AES-128`

(including a data pair and a key value conforming the inbound parts from Δz_1 to Δw_5) is 2^8. The probablity of the outbound phase is 2^{-64}, we need 2^{64} starting points to find one collision. In other words, given 21 bytes of Δz_1, Δz_2, Δw_4, Δw_5, $\Delta w_3[0,2]$, $\Delta w_3[4,5]$, $\Delta w_3[8,9]$, $\Delta w_3[13,14]$ and $k_3[6]$ $k_4[0]$, $k_4[4]$, we get one collision with probability of 2^{-72}. Since $21 \times 8 = 168 > 72$, we have enough degrees of freedom to find one collision.

The Quantum Attack. Given input-output differences of the active Sbox, it is expected to find one pair on average, e.g. (x, x'), satisfying the differences. In inbound phase of the quantum rebound attacks, given a valid input-output differences of l active Sboxes, we obtain about $2^l/2$ choices for starting points. To indicate which starting point to choose among $2^l/2$ choices, Hosoyamda and

1. $w_3[0] = k_4[0] \oplus x_4[0]$	23. $k_3[10] = k_4[6] \oplus k_4[10]$
2. $w_3[2,3] = \texttt{MC}^{-1}(w_3[0], z_3[0,2,3])$	24. $k_5[4] = k_5[0] \oplus k_4[4]$
3. $k_4[2,3] = w_3[2,3] \oplus x_4[2,3]$	25. $w_3[10] = x_4[10] \oplus k_4[10]$
4. $k_4[13] = \texttt{SB}^{-1}(k_4[0] \oplus k_5[0] \oplus r_c)$	26. $w_3[8,9] = \texttt{MC}^{-1}(w_3[10], z_3[8,9,10])$
5. $w_3[13] = x_4[13] \oplus k_4[13]$	27. $k_4[8,9] = w_3[8,9] \oplus x_4[8,9]$
6. $w_3[14,15] = \texttt{MC}^{-1}(w_3[13], z_3[12,13,15])$	28. $k_3[9] = k_4[5] \oplus k_4[9]$
7. $k_4[14,15] = w_3[14,15] \oplus x_4[14,15]$	29. $k_3[8] = k_4[4] \oplus k_4[8]$
8. $k_5[14] = k_5[10] \oplus k_4[14]$	30. $w_2[8,9,10] = x_3[8,9,10] \oplus k_3[8,9,10]$
9. $k_4[6] = k_4[2] \oplus k_3[6]$	31. $w_2[11] = \texttt{MC}^{-1}(w_2[8,9,10], z_2[10])$
10. $w_3[4] = x_4[4] \oplus k_4[4]$	32. $k_3[11] = w_2[11] \oplus x_3[11]$
11. $w_3[5,7] = \texttt{MC}^{-1}(w_3[4], z_3[5,6,7])$	33. $k_4[11] = k_4[7] \oplus k_3[11]$
12. $k_4[5,7] = w_3[5,7] \oplus x_4[5,7]$	34. $k_3[13] = k_4[9] \oplus k_4[13]$
13. $k_3[4] = k_4[0] \oplus k_4[4]$	35. $k_3[15] = k_4[11] \oplus k_4[15]$
14. $k_3[7] = k_4[3] \oplus k_4[7]$	36. $w_2[13,14,15] = k_3[13,14,15] \oplus x_3[13,14,15]$
15. $k_5[1] = k_4[5] \oplus k_5[5]$	37. $w_2[12] = \texttt{MC}^{-1}(w_2[13,14,15], z_2[13])$
16. $k_4[1] = k_5[1] \oplus \texttt{SB}(k_4[14]) \oplus r_c$	38. $k_3[12] = x_3[12] \oplus w_2[12]$
17. $k_3[5] = k_4[5] \oplus k_4[1]$	39. $k_3[0] = \texttt{SB}(k_3[13]) \oplus k_4[0] \oplus r_c$
18. $k_5[2] = k_4[2] \oplus \texttt{SB}(k_4[15]) \oplus r_c$	40. $k_3[2] = \texttt{SB}(k_3[15]) \oplus k_4[2] \oplus r_c$
19. $k_5[6] = k_5[2] \oplus k_4[6]$	41. $w_2[0,1,2] = x_3[0,1,2] \oplus k_3[0,1,2]$
20. $k_4[10] = k_5[6] \oplus k_5[10]$	42. $w_2[3] = \texttt{MC}^{-1}(w_2[0,1,2], z_2[0])$
21. $k_3[14] = k_4[10] \oplus k_4[14]$	43. $k_3[3] = w_2[3] \oplus x_3[3] \stackrel{?}{=} \texttt{SB}(k_3[12]) \oplus k_4[3] \oplus r_c$
22. $k_3[1] = \texttt{SB}(k_3[14]) \oplus k_4[1] \oplus r_c$	

Table 3: Steps to recover the subkey k_3 from known bytes

Sasaki [39, Sect. 6.2, Page 22] introduce $l-1$ auxiliary qubits. In Hosoyamda and Sasaki's attack, l is usually small, e.g. $l=4$, then the increased time complexity due to the $l-1$ auxiliary qubits is marginal. However, in our 8-round attack of Figure 10, we have to compute pairs for $l = 3+3+12+12 = 30$ active Sboxes with given input-output differences. If we follow Hosoyamda and Sasaki's idea [39], we have to introduce $l-1 = 29$ auxiliary qubits, which can not be ignored anymore. To deal with the problem, we introduce a trick in Supplementary Material A, which introduces the auxiliary qubits nearly for free. By our new trick, we perform the 8-round quantum semi-free-start collision attack. According to Equation 5, we have $l = 3+3+12+12 = 30$, and choose 9 bytes out of the 21 bytes to act as "$|\Delta_{in}| + |\Delta_{out}|$" to find the collision with time complexity roughly $30 \cdot 2^{72/2}/128 \approx 2^{34}$ 8-round AES, which is roughly the square root of 2^{72}.

Practical Semi-Free-Start Collision Attack on 6-Round AES-128. To prove the correctness of the 8-round semi-free-start collision, we give a 6-round practical semi-free-start collision by cutting the differential characteristic depicted in Figure 10 to 6 rounds (round 1 to 6). The total complexity to find the collision is 2^{24}, including 2^8 time for solving the Super-Inbound phase and 2^{16} for the condition $\Delta x_1 = \Delta z_6$. We have verified the validity and complexity of the attack by implementing the 6-round attack. One such example collision is presented in Table 4 in Supplementary Material C.

4.3 Quantum Collision Attack on 8-Round AES-128

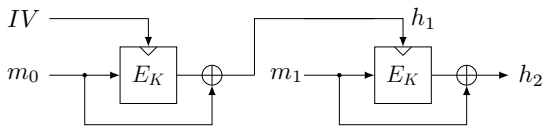


Fig. 11: Matyas-Meyer-Oseas (MMO) hashing mode with two blocks

Remove the Inbound I/II from Figure 10, we get the trail for 8-round collision attack (also shown in Figure 15). The inbound phase covers states z_2 to w_4. The probability of the outbound phase is 2^{-96}. For a random given Δz_2, Δw_4 and the master key, the steps to get one starting point are as follows:

1. Deduce Δx_3 and Δy_4 from Δz_2 and Δw_4.
2. For $\texttt{SSB}^{(0)}$ marked in red, randomly choose $\Delta w_3[0,2] \in \mathbb{F}_2^{16}$,
   (a) Compute other active differences $\Delta z_3[0,2,3]$ and $\Delta w_3[3]$ by Property 1
   (b) Deduce the values for active bytes by accessing DDT to get $(z_3[0,2,3], z_3'[0,2,3])$ and $(x_4[0,2,3], x_4'[0,2,3])$
   (c) Deduce $(w_4[0,2,3], w_4'[0,2,3])$ by XORing $(x_4[0,2,3], x_4'[0,2,3])$ with k_4
   (d) Check if $\texttt{MC}(z_3[0,2,3]) \stackrel{?}{=} w_4[0,2,3]$, which is a 16-bit filter by Pro. 1
3. Repeat Step 2 in parallel for $\texttt{SSB}^{(1)}$, $\texttt{SSB}^{(2)}$, and $\texttt{SSB}^{(3)}$ to get one starting point on average.

Hence, we need 2^{16} time complexity to get one starting point for a random given a random given Δz_2, Δw_4 and the master key. Since the probability of the outbound phase is 2^{-96}, we need 2^{96} starting points to get one collision. Since $|\Delta z_2| \times |\Delta w_4| = 2^{48} < 2^{96}$, we need two blocks as shown in Figure 11 to get additional degrees of freedom from the first block. The overall time complexity of the classical time is 2^{112}. According to the quantum analysis by [39,29], we apply Grover's algorithm and get a quantum collision attack with roughly 2^{56} quantum time complexity. Supplementary Material B gives a very detailed quantum collision attack on 8-round AES, whose time complexity is $2^{55.53}$ 8-round AES, which is very close to the estimated 2^{56}.

5 Improved Quantum Attacks on Saturnin-Hash

Saturnin is a block cipher with a 256-bit state and 256-bit key that was designed as the derivative of AES with efficient implementation by Canteaut et al. [17]. It is among the round 2 candidates of the NIST lightweight cryptography competition. The composition of two consecutive rounds starting from even round is called super-round, which is very similar to an AES round operating on 16-bit words except that the SR is replaced by a transposition. Saturnin-Hash is built on 16-super-round Saturnin block cipher with the MMO hashing mode.

5.1 Improved 8-Round Quantum Free-Start Collision

The differential trail we use here was found in [30]. As shown in Figure 12, our new method extends the inbound phase by covering round 0 to round 4 in inbound phase, and the probability of the outbound phase becomes $2^{-135.8}$.

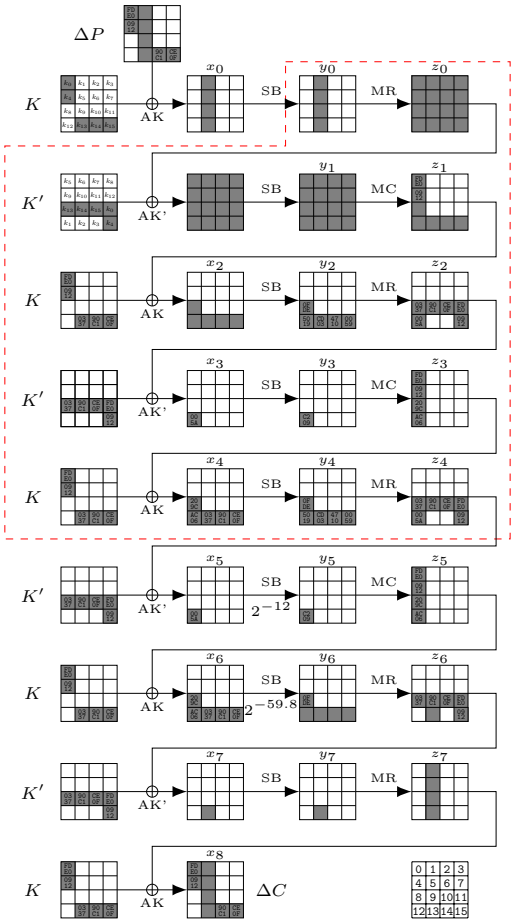


Fig. 12: Free-start collision attack on 8-round `Saturnin-Hash`

Super-Inbound Phase. In Dong et al.'s [30] attack, the inbound phase covers states from y_0 to x_3. However, our Super-Inbound covers two more rounds from state y_0 to x_5. Hence, the probabilities $Pr(\Delta x_3 \mapsto \Delta y_3) = 2^{-12}$ and $Pr(\Delta x_4 \mapsto \Delta z_4) = 2^{-59.8}$ in Dong et al.'s attack are removed in ours.

To perform the 4-round Super-Inbound phase, we first randomly assign arbitrary differences for all unfixed differences in $\Delta z_0 = \texttt{MR}(\Delta y_0)$, Δz_1 and Δz_4.

Together with the prefixed differences of Δz_2 and Δz_3, we do the following steps to find a pair to confirm the given differences in the Super-Inbound phase.

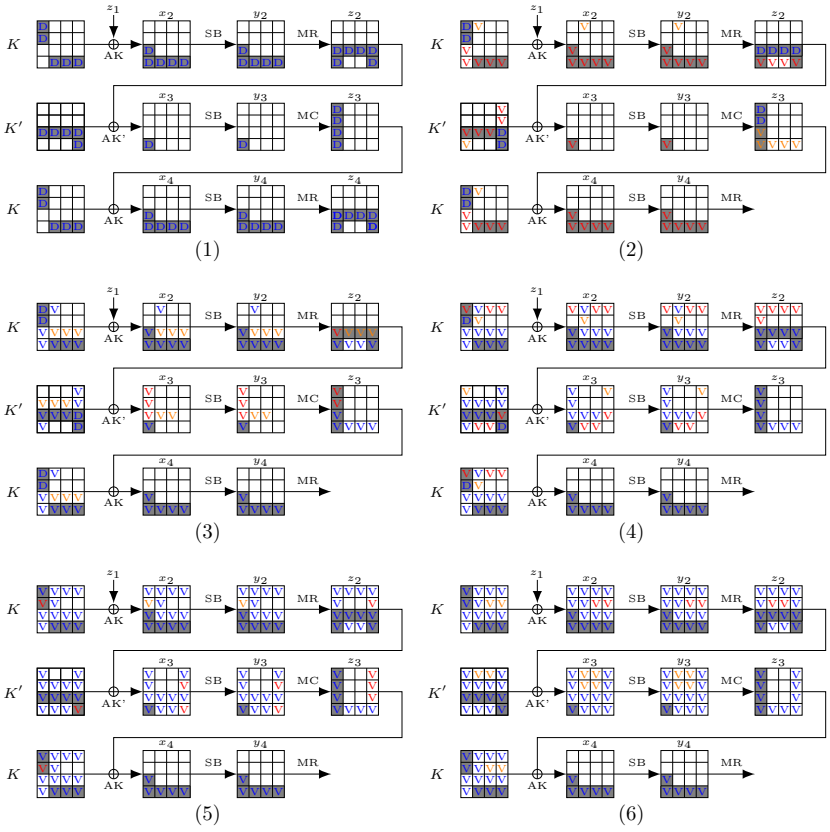


Fig. 13: Solving the inbound part for 8-round `Saturnin-Hash`

1. Deduce the differences of Δy_1, Δy_4. Compute all the active byte values of $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ and (x_4, y_4) by assessing the `DDT`. Then z_1 is known. As shown in Figure 13, we pick out round 2 to round 4 to clarify the steps to compute a conforming pair for the inbound part. As shown Figure 13 (2), compute $K[V] = z_1 \oplus x_2[V]$. Compute $K'[V] = x_3[V] \oplus z_2[V]$. Similarly, compute all other cells marked by "V" and "V".
2. In Figure 13 (3), deduce "V" and "V" cells by the "V" cells. **Guess** $z_3[0]$ to compute $y_3[V]$ and $z_3[V]$ by Property 1. Compute $z_2[V] = K'[V] \oplus x_3[V]$. **Guess** $z_2[9, 10]$ to deduce $z_2[V]$ and $y_2[V]$ by Property 1. Then, compute $K[V] = x_2[V] \oplus z_1$ and $x_3[V] = z_2[V] \oplus K'[V]$.
3. In Figure 13 (4), **guess** $z_2[0, 1, 2]$ to deduce $y_2[V]$ and $z_2[V]$. Deduce $K[V]$ and $K'[V]$. Compute $K'[V] = x_3[V] \oplus z_2[V]$.

4. In Figure 13 (5), **guess** $z_3[3]$ to deduce $z_3[V]$ and $y_2[V]$. Compute $K'[V] = x_3[V] \oplus z_2[V]$ and $z_2[V] = K'[V] \oplus x_3[V]$. Compute $x_2[V] = K[V] \oplus z_1$.
5. In Figure 13 (6), compute $z_2[V]$ and $y_2[V]$. Compute $K[V] = x_2[V] \oplus z_1$. Compute $y_3[V]$, so that the 2nd column and 3rd column of y_3 and z_3 form a filter of 2^{-32}.

Totally, we guess 7-cell $(z_3[0,3], z_2[9,10,0,1,2])$ to deduce the conforming pair. In Step 5, there is a filter of 2^{-32}, together with the outbound probability $2^{-135.8}$, we get a collision with total probability of $2^{-32-135.8} = 2^{167.8}$. We have $2^{9\times16} = 2^{144}$ choices for $\Delta z_0 = \texttt{MR}(\Delta y_0)$, Δz_1 and Δz_4. Together with the 7-cell $(z_3[0,3], z_2[9,10,0,1,2])$, the total degree of freedom is $2^{144+16\times7} = 2^{256} > 2^{167.8}$. Hence, we have enough degree of freedom to find the collision. Since we do not have the quantum circuit for the `DDT` of `Saturnin` like `AES`, we have to use Equation 4 to estimate the time complexity. With $l = 27$, the time is roughly $27 \cdot 2^{167.8/2+16/2}/(16 \times 8) = 2^{89.65}$ 8-round `Saturnin-Hash`.

5.2 Extend the Attack to 10-round Free-Start Collision

The 10-round differential trail is easily obtained by repeating the path of Figure 12 round 5 and round 6 one more time before entering round 7. The figure for 10-round `Saturnin-Hash` is given in Figure 17 in Supplementary Material E. By this way, the probability of the outbound phase decreases by $2^{-16-59.8} = 2^{-75.8}$ while the Super-Inbound phase maintains the same probability 2^{-32}. Finally, we obtain the free-start collision attack with probability $2^{167.8-75.8} = 2^{-243.6}$. Since the number of degrees of freedom 2^{256} is still larger than $2^{243.6}$, so that we can find a collision. According to Equation 4, $l = 27$, the time is roughly $27 \cdot 2^{243.6/2+16/2}/(16 \times 10) = 2^{127.2}$ 10-round `Saturnin-Hash`. Additionally, we present an improved 7-round quantum semi-free-start collision in Supplementary Material D.

6 Quantum Collision Attack on `SKINNY-128-384-MMO`

`SKINNY` is a family of lightweight block ciphers designed by Beierle et al. [7]. In this section, we focus on the hashing mode of `SKINNY`-128-384. Please find the structure of `SKINNY`-n-$3n$ in [7]. The `MC` operation is non-MDS:

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

6.1 21-Round Quantum Free-Start Collision Attack

In quantum setting, we derive the free-start collision attack on hashing modes (MMO/MP) with 21-round `SKINNY-128-384` using the differential characteristic shown in Figure 18 in Supplementary Material F. The new differential trail is

found by using the automatic tool in [23]. The 5-round Super-Inbound covers from state w_9 to x_{14}. The outbound phase happens with probability $2^{-103.4}$.

We pick out the 5-round Super-Inbound phase from the whole rebound attack trail of Figure 18 to get Figure 14 (a). To launch the rebound attack, we first precompute a pair satisfying the 5-round inbound trail as shown in Figure 14 (b). Then thanks to the large degrees of freedom from the tweakey, we get enough starting points by changing the 384-bit tweakey.

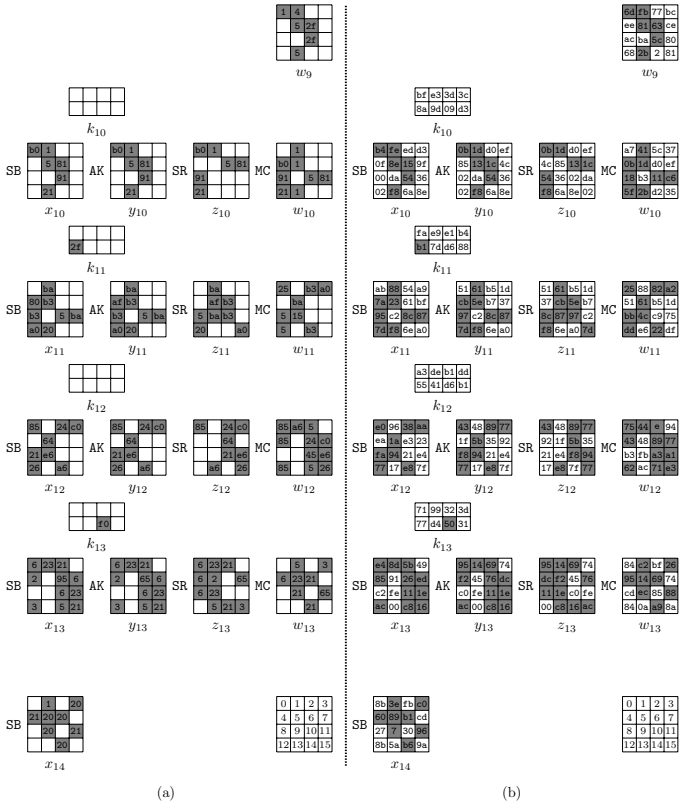


Fig. 14: 5-round Super-Inbound of SKINNY-128-384: (a) Differences, (b) Values. The values of k_i are the XOR of subkeys and constants of AC operator.

Precomputation in the Super-Inbound Phase. We build several steps to compute the conforming data pairs for the Super-Inbound phase.

1. In Figure 14(a), focusing on states z_{10} to x_{11}, the values in $z_{10}[8, 12] = \mathtt{SR}(x_{10}[10, 13])$ and all active bytes of w_{10} are deduced by assessing DDT with fixed differences in Figure 14(a). In the 1st column z_{10} and w_{10}, i.e., $z_{10}^{(0)}$ and $w_{10}^{(0)}$, we have a condition "$w_{10}[4] \oplus w_{10}[12] = z_{10}[8]$" due to Equation 1, which is a filter of 2^{-8}. In our trail, the differences are dedicatedly

chosen, so that we have enough pairs to verify the filters for the given differences. For example, in the condition "$w_{10}[4] \oplus w_{10}[12] = z_{10}[8]$", we choose $(w_{10}[4], w_{10}[12], z_{10}[8]) \in (\texttt{DDT}[\texttt{b0}_\texttt{x}][\texttt{80}_\texttt{x}] \times \texttt{DDT}[\texttt{21}_\texttt{x}][\texttt{a0}_\texttt{x}] \times \texttt{DDT}[\texttt{2f}_\texttt{x}][\texttt{91}_\texttt{x}])$, where $\texttt{DDT}[\texttt{b0}_\texttt{x}][\texttt{80}_\texttt{x}]$ is the subset of DDT with input-output differences $(\texttt{b0}_\texttt{x}, \texttt{80}_\texttt{x})$. The size $|\texttt{DDT}[\texttt{b0}_\texttt{x}][\texttt{80}_\texttt{x}] \times \texttt{DDT}[\texttt{21}_\texttt{x}][\texttt{a0}_\texttt{x}] \times \texttt{DDT}[\texttt{2f}_\texttt{x}][\texttt{91}_\texttt{x}]| = 48 \cdot 2^5 \cdot 2^3 > 2^8$.
2. With the choice of $w_{10}[12], z_{11}[15]$ is known as well. $z_{11}[8, 9, 10, 12]$ and all active bytes of w_{11} are deduced by DDT. We have similar conditions to fulfill for the 1st column of z_{11} and w_{11} with $w_{11}[0] \oplus w_{11}[12] = z_{11}[12]$, which act as a filter of 2^{-8}.
3. Do similar steps from z_{10} to w_{13}, we get a data and key pair as shown in Figure 14(b) conforming the whole 5-round inbound trail.

An additional trick is used in our inbound phase. As only the first two rows of x_9 are XORed with k_9, the last two rows of x_9 and w_8 are fully determined by w_9. Hence, in the inbound phase, we directly find the pair with $\Delta w_8[8] = \texttt{04}_\texttt{x}, \Delta w_8[13] = \texttt{54}_\texttt{x}$, and $\Delta w_8[14] = \texttt{54}_\texttt{x}$, which increases the probability of the outbound phase from $2^{-103.4}$ to $2^{-92.4}$. The conforming pair for the Super-Inbound is given in Figure 14(b), which is generated by the source code at https://www.dropbox.com/s/2n4d9zwufkpigqt/Find_inbound.7z

Generating Starting Points for Outbound Phase. Now we have the values of subkeys $k_{10}, k_{11}, k_{12}, k_{13}$. Since the key schedule is linear and by solving a linear system on the 384-bit key with fixed values of subkeys $k_{10}, k_{11}, k_{12}, k_{13}$ in Figure 14 (b), we can derive at most 2^{128} starting points for the outbound phase. We need only $2^{92.4}$ starting points to get a collision. In quantum setting, the $2^{92.4}$ starting points are traversed by Grover's algorithm, which need roughly $2^{46.2}$ time complexity to find the collision.

6.2 Classic Free-Start Collision Attack on 19-Round

The first classical free-start collision attack on 19-round SKINNY-128-384 hash mode is also given with the trail in Figure 19 in Supplementary Material F. The Super-Inbound covers 5 rounds from round 9 to round 13, and only one data pair, shown in Figure 20 in Supplementary Material F, is needed to confirm the inbound trail in the precomputation phase. Similar to the 21-round attack, the pre-computed pair of the inbound phase also satisfies the differential of the last two rows of x_8 to w_7. Since the outbound excluding the Sboxes in the last two rows of x_8 happens with probability $2^{-51.2}$, the final time complexity is $2^{51.2}$.

7 Discussion and Conclusion

7.1 Possible Generalization of Triangulating Rebound

In Section 3, we have shown that the multiple inbound phases can be connected by the key bytes. We may describe the idea in a more generalized way. In fact, for active Sbox or Super-Sbox with fixed input/output differences, the values are fixed. For active Sbox without fixed input/output differences, or inactive Sbox, the values are not fixed and up to attacker's control, which are the source

of the degree of freedom (DoF). In order to identify a data pair conforming the truncated differential, we may consume the degree of freedom from the unfixed Sbox to bridge several fixed values due to active Sboxes or Super-Sboxes with fixed input/output differences. This intuitive idea implies that not only the key bytes are useful to connect the differences, but also the internal states. In Supplementary Material G, we give an example to connect the differences with DoF from internal states and gain some improved collision attacks on Grøstl.

7.2 Conclusion

In this paper, we extended the number of attacked rounds by the rebound attack by introducing *triangulating rebound attack*. The core idea is to take full advantage of the available degrees of freedom from both the subkeys and the state to greatly extend the number of rounds covered by the inbound phase, which is named as Super-Inbound. As a consequence, multiple improved classic or quantum collision attacks on AES-like hashing were presented. Possible future works are to investigate more ciphers with our method, as well as to find more applications such as distinguishing attacks based the improved rebound attacks.

Acknowledgments

We would like to thank the anonymous reviewers from EUROCRYPT 2022 and CRYPTO 2022 for their valuable comments. This research is partially supported by Nanyang Technological University in Singapore under Grant 04INS000397C230, and Singapore's Ministry of Education under Grants RG91/20 and MOE2019-T2-1-060. Xiaoyang Dong is supported by National Key R&D Program of China (2018YFA0704701), the Major Program of Guangdong Basic and Applied Research (2019B030302008), Major Scientific and Technological Innovation Project of Shandong Province, China (2019JZZY010133), Natural Science Foundation of China (61902207).

References

1. Z. Alliance. ZigBee 2007 specification. *Online: http://www.zigbee.org/*, 2007.
2. G. Banegas and D. J. Bernstein. Low-communication parallel quantum multi-target preimage search. In *SAC 2017, Revised Selected Papers*, volume 10719, pages 325–335. Springer.
3. Z. Bao, L. Ding, J. Guo, H. Wang, and W. Zhang. Improved meet-in-the-middle preimage attacks against AES hashing modes. *IACR Trans. Symmetric Cryptol.*, 2019(4):318–347, 2019.
4. Z. Bao, X. Dong, J. Guo, Z. Li, D. Shi, S. Sun, and X. Wang. Automatic search of meet-in-the-middle preimage attacks on AES-like hashing. In *EUROCRYPT 2021, Proceedings, Part I*, volume 12696, pages 771–804. Springer.
5. Z. Bao, J. Guo, S. Li, and P. Pham. Quantum Multi-Collision Distinguishers. Cryptology ePrint Archive, Report 2021/703, 2021. https://ia.cr/2021/703.

6. P. S. Barreto and V. Rijmen. The WHIRLPOOL hashing function. *Submitted to NESSIE*.
7. C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *CRYPTO 2016, Proceedings, Part II*, pages 123–153. Springer, 2016.
8. R. Benadjila, O. Billet, H. Gilbert, G. Macario-Rat, T. Peyrin, M. Robshaw, and Y. Seurin. SHA-3 proposal: ECHO. *Submission to NIST (updated)*, page 113, 2009.
9. D. J. Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete. *SHARCS 2009 9: 105*.
10. G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3(30):320–337, 2009.
11. A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique cryptanalysis of the full AES. In *ASIACRYPT 2011, Proceedings*, volume 7073, pages 344–371. Springer.
12. X. Bonnetain, A. Hosoyamada, M. Naya-Plasencia, Y. Sasaki, and A. Schrottenloher. Quantum attacks without superposition queries: The offline Simon's algorithm. In *ASIACRYPT 2019, Proceedings, Part I*, pages 552–583.
13. X. Bonnetain, M. Naya-Plasencia, and A. Schrottenloher. On quantum slide attacks. In *SAC 2019, Waterloo, ON, Canada, August 12-16, 2019*, pages 492–519, 2019.
14. X. Bonnetain, M. Naya-Plasencia, and A. Schrottenloher. Quantum security analysis of AES. *IACR Trans. Symmetric Cryptol.*, 2019(2):55–93, 2019.
15. G. Brassard, P. Høyer, and A. Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN '98, Campinas, Brazil, April, 20-24, 1998, Proceedings*, pages 163–169, 1998.
16. A. Canteaut, S. Duval, G. Leurent, M. Naya-Plasencia, L. Perrin, T. Pornin, and A. Schrottenloher. A note on related-key attacks on Saturnin. https://project.inria.fr/saturnin/files/2020/11/Note-RK-1.pdf.
17. A. Canteaut, S. Duval, G. Leurent, M. Naya-Plasencia, L. Perrin, T. Pornin, and A. Schrottenloher. Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. *IACR Trans. Symmetric Cryptol.*, 2020(S1):160–207, 2020.
18. V. Cauchois, C. Gomez, and R. Lercier. Grøstl distinguishing attack: A new rebound attack of an aes-like permutation. *IACR Trans. Symmetric Cryptol.*, 2017(3):1–23, 2017.
19. A. Chailloux, M. Naya-Plasencia, and A. Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *ASIACRYPT 2017, Proceedings, Part II*, pages 211–240, 2017.
20. J. Daemen and V. Rijmen. Understanding two-round differentials in AES. In *SCN 2006, Proceedings*, volume 4116, pages 78–94. Springer.
21. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
22. I. Damgård. A design principle for hash functions. In *CRYPTO '89, Proceedings*, pages 416–427, 1989.
23. S. Delaune, P. Derbez, and M. Vavrille. Catching the fastest boomerangs. *IACR Transactions on Symmetric Cryptology*, pages 104–129, 2020.
24. P. Derbez, P. Fouque, and J. Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In *EUROCRYPT 2013, Proceedings*, volume 7881, pages 371–387.

25. P. Derbez, P. Huynh, V. Lallemand, M. Naya-Plasencia, L. Perrin, and A. Schrottenloher. Cryptanalysis results on Spook - bringing full-round Shadow-512 to the light. In *CRYPTO 2020, Proceedings, Part III*, volume 12172, pages 359–388.
26. I. Dinur, O. Dunkelman, N. Keller, and A. Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *CRYPTO 2012, Proceedings*, volume 7417, pages 719–740. Springer.
27. X. Dong, B. Dong, and X. Wang. Quantum attacks on some Feistel block ciphers. *Des. Codes Cryptogr.*, 88(6):1179–1203, 2020.
28. X. Dong, J. Hua, S. Sun, Z. Li, X. Wang, and L. Hu. Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In *CRYPTO 2021, Proceedings, Part III*, volume 12827, pages 278–308.
29. X. Dong, S. Sun, D. Shi, F. Gao, X. Wang, and L. Hu. Quantum collision attacks on AES-like hashing with low quantum random access memories. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020, Proceedings, Part II*, volume 12492, pages 727–757.
30. X. Dong, Z. Zhang, S. Sun, C. Wei, X. Wang, and L. Hu. Automatic classical and quantum rebound attacks on AES-like hashing by exploiting related-key differentials. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 241–271. Springer, 2021.
31. A. Duc, J. Guo, T. Peyrin, and L. Wei. Unaligned rebound attack: Application to Keccak. In *FSE 2012, Revised Selected Papers*, volume 7549, pages 402–421.
32. O. Dunkelman, N. Keller, and A. Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. In *ASIACRYPT 2010, Proceedings*, volume 6477, pages 158–176. Springer.
33. P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. Grøstl - a SHA-3 candidate. In *Symmetric Cryptography, 11.01. - 16.01.2009*, 2009.
34. H. Gilbert and T. Peyrin. Super-Sbox cryptanalysis: Improved attacks for AES-like permutations. In *FSE 2010, Seoul, Korea, February 7-10, 2010*, pages 365–383, 2010.
35. L. Grassi, M. Naya-Plasencia, and A. Schrottenloher. Quantum algorithms for the k-xor problem. In *ASIACRYPT 2018, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, pages 527–559, 2018.
36. L. 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.
37. C. Guo, J. Katz, X. Wang, and Y. Yu. Efficient and Secure Multiparty Computation from Fixed-Key Block Ciphers. In *2020 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, May 18-21, 2020*, pages 825–841, 2020.
38. A. Hosoyamada and Y. 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.
39. A. Hosoyamada and Y. Sasaki. Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Proceedings, Part II*, volume 12106, pages 249–279.
40. A. Hosoyamada and Y. Sasaki. Quantum collision attacks on reduced SHA-256 and SHA-512. In *CRYPTO 2021*, volume 12825, pages 616–646. Springer.

41. A. Hosoyamada, Y. Sasaki, S. Tani, and K. Xagawa. Improved quantum multicollision-finding algorithm. In *PQCrypto 2019, Revised Selected Papers*, volume 11505, pages 350–367. Springer.
42. A. Hosoyamada, Y. Sasaki, and K. Xagawa. Quantum multicollision-finding algorithm. In *ASIACRYPT 2017, Proceedings, Part II*, volume 10625, pages 179–210.
43. ISO/IEC. 10118-2:2010 Information technology — Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher. 3rd ed., International Organization for Standardization, Geneve, Switzerland, October, 2010.
44. J. Jean and P. Fouque. Practical near-collisions and collisions on round-reduced ECHO-256 compression function. In A. Joux, editor, *FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733, pages 107–127.
45. J. Jean, M. Naya-Plasencia, and T. Peyrin. Improved rebound attack on the finalist Grøstl. In *FSE 2012, Washington, DC, USA, March 19-21, 2012*, pages 110–126, 2012.
46. J. Jean, M. Naya-Plasencia, and T. Peyrin. Multiple limited-birthday distinguishers and applications. In *SAC 2013, Burnaby, BC, Canada, August 14-16, 2013*, pages 533–550, 2013.
47. M. Kaplan, G. Leurent, A. Leverrier, and M. 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.
48. M. Kaplan, G. Leurent, A. Leverrier, and M. Naya-Plasencia. Quantum differential and linear cryptanalysis. *IACR Trans. Symmetric Cryptol.*, 2016(1):71–94, 2016.
49. M. Keller, E. Orsini, and P. Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 830–842, 2016.
50. D. Khovratovich, A. Biryukov, and I. Nikolic. Speeding up collision search for byte-oriented hash functions. In *CT-RSA 2009, Proceedings*, volume 5473, pages 164–181.
51. D. Khovratovich, I. Nikolic, and C. Rechberger. Rotational rebound attacks on reduced Skein. *J. Cryptol.*, 27(3):452–479, 2014.
52. L. R. Knudsen, C. Rechberger, and S. S. Thomsen. The Grindahl hash functions. In *FSE 2007, Revised Selected Papers*, volume 4593, pages 39–57.
53. S. Kölbl, M. M. Lauridsen, F. Mendel, and C. Rechberger. Haraka v2 - efficient short-input hashing for post-quantum applications. *IACR Trans. Symmetric Cryptol.*, 2016(2):1–29, 2016.
54. H. Kuwakado and M. 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.
55. H. Kuwakado and M. Morii. Security on the quantum-type Even-Mansour cipher. In *ISITA 2012, Honolulu, HI, USA, October 28-31, 2012*, pages 312–316, 2012.
56. M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen, and M. Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In *ASIACRYPT 2009, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 126–143, 2009.
57. G. Leander and A. May. Grover Meets Simon - quantumly attacking the FX-construction. In *ASIACRYPT 2017, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, pages 161–178, 2017.
58. Q. Liu and M. Zhandry. On finding quantum multi-collisions. In *EUROCRYPT 2019, Proceedings, Part III*, volume 11478, pages 189–218. Springer.

59. K. Matusiewicz, M. Naya-Plasencia, I. Nikolic, Y. Sasaki, and M. Schläffer. Rebound attack on the full LANE compression function. In *ASIACRYPT 2009, Proceedings*, volume 5912, pages 106–125.
60. F. Mendel, T. Peyrin, C. Rechberger, and M. Schläffer. Improved cryptanalysis of the reduced grøstl compression function, ECHO permutation and AES block cipher. In *SAC 2009, Revised Selected Papers*, volume 5867, pages 16–35. Springer.
61. F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. Rebound attacks on the reduced Grøstl hash function. In *CT-RSA 2010, Proceedings*, volume 5985, pages 350–365.
62. F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *FSE 2009, Leuven, Belgium, February 22-25, 2009*, pages 260–276, 2009.
63. F. Mendel, V. Rijmen, and M. Schläffer. Collision attack on 5 rounds of Grøstl. In *FSE 2014, London, UK, March 3-5, 2014*, pages 509–521, 2014.
64. A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
65. R. C. Merkle. A certified digital signature. In *CRYPTO '89, 1989, Proceedings*, pages 218–238, 1989.
66. M. Naya-Plasencia. How to improve rebound attacks. In *CRYPTO 2011, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 188–205, 2011.
67. M. Naya-Plasencia and A. Schrottenloher. Optimal merging in quantum k-xor and k-xor-sum algorithms. In *EUROCRYPT 2020, Proceedings, Part II*, volume 12106, pages 311–340. Springer.
68. M. Naya-Plasencia, D. Toz, and K. Varici. Rebound attack on JH42. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011, Proceedings*, volume 7073, pages 252–269. Springer.
69. Y. Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool. In *FSE 2011, Revised Selected Papers*, pages 378–396.
70. Y. Sasaki, Y. Li, L. Wang, K. Sakiyama, and K. Ohta. Non-full-active Super-Sbox analysis: Applications to ECHO and grøstl. In *ASIACRYPT 2010, Singapore, December 5-9, 2010. Proceedings*, pages 38–55, 2010.
71. Y. Sasaki, L. Wang, S. Wu, and W. Wu. Investigating fundamental security requirements on Whirlpool: Improved preimage and collision attacks. In X. Wang and K. Sako, editors, *ASIACRYPT 2012, Proceedings*, volume 7658, pages 562–579.
72. M. Schläffer. Updated differential analysis of Grøstl. *Grøstl website (January 2011)*, 2011.
73. P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134, 1994.
74. M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. The first collision for full SHA-1. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Proceedings, Part I*, volume 10401, pages 570–596. Springer, 2017.
75. P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptol.*, 12(1):1–28, 1999.
76. X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full SHA-1. In V. Shoup, editor, *CRYPTO 2005, Proceedings*, volume 3621, pages 17–36. Springer, 2005.
77. X. Wang and H. Yu. How to break MD5 and other hash functions. In R. Cramer, editor, *EUROCRYPT 2005, Proceedings*, volume 3494, pages 19–35. Springer, 2005.

Supplementary Material

A Quantum Rebound Attack with Quantum Amplitude Amplification

The quanutm oracle of a function $f : \mathbb{F}_2^m \mapsto \mathbb{F}_2^n$ is the unitary operator $\mathcal{U}_f$ that $\mathcal{U}_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$ with $x \in \mathbb{F}_2^m$ and $y \in \mathbb{F}_2^n$.

Variations on Grover's Algorithm. The task is to find a marked element from a set X. We denote by $M \subset X$ the subset of marked elements and assume that we know the fraction $\epsilon = |M|/|X|$ of marked elements. A classical algorithm to solve this problem is to repeat $O(1/\epsilon)$ times. The quantum algorithm can therefore be expressed as a function of two parameters:

- The *Setup* operation, which is to sample a uniform element from X. Denote the *Setup* cost (running time) as $|Setup|_{RT}$.
- The *Checking* operation, which is to check if an element is marked. Denote the *Checking* cost (running time) as $|Checking|_{RT}$.

The cost of the quantum algorithm can be the time or the number of queries to the input. It suffices to consider specifically one of those resources when quantifying the *Setup* and *Checking* cost. Similarly, Grover's algorithm [36] is a quantum search procedure that finds a marked element, and whose complexity can be written as a function of the quantum *Setup* cost $|Setup|_{RT}$, which is the cost of constructing a uniform superposition of all elements in X, and the quantum *Checking* cost $|Checking|_{RT}$, which is the cost of applying a controlled-phase gate to the marked elements. Notice that a classical or a quantum algorithm that checks membership to M can easily be modified to get a controlled-phase. The time complexity of Grover's algorithm is $\sqrt{1/\epsilon} \cdot (|Setup|_{RT} + |Checking|_{RT})$. Assuming that the *Setup* and *Checking* procedures are easy, Grover's algorithm can then find an element $x \in M$ a cost $\sqrt{1/\epsilon}$.

Grover's algorithm can also be written as a special case of quantum amplitude amplification (QAA), a quantum technique introduced by Brassard, Høyer and Tapp in order to boost the success probability of quantum algorithms [15]. Intuitively, assume that a quantum algorithm $\mathcal{A}$ produces a superposition of outputs in a good subspace $G \subset X$ and outputs in a bad subspace $B \subset X$. Then there exists a quantum algorithm that calls $\mathcal{A}$ as a subroutine to amplify the amplitude of good outputs. Suppose $\mathcal{A}$ was a classical algorithm, repeating it $O(1/a)$ times to produce a good output, where a is the probability of producing a good output. Just as Grover's algorithm, the quantum amplitude amplification (QAA) technique achieves the same result with a quadratic improvement. The time complexity of QAA is about

$$\sqrt{1/a} \cdot (|\mathcal{A}|_{RT} + |Checking|_{RT}). \tag{2}$$

Hosoyamda and Sasaki [39]'s Quantum Rebound Attack With Auxiliary Variables. At EUROCRYPT 2020, Hosoyamda and Sasaki [39] applied Grover's algorithm to the rebound attack. In the inbound phase, we are going to compute the pair (S, S'), i.e. the so-called starting point, for the active Sboxes or Super-Sboxes that satisfying the inbound differential trail. Suppose the state S consists of l active Sboxes (denoted as $\mathtt{SB}^{(i)}$ with $1 \leq i \leq l$) with arbitrary non-zero input/output differences $(\delta_{in}^{(i)}, \delta_{out}^{(i)})$ $(1 \leq i \leq l)$ in parallel. Let $\Delta_{in} = \delta_{in}^{(1)} \| \delta_{in}^{(2)} \| \cdots \| \delta_{in}^{(l)}$ and $\Delta_{out} = \delta_{out}^{(1)} \| \delta_{out}^{(2)} \| \cdots \| \delta_{out}^{(l)}$. Randomly given a value for $(\Delta_{in}, \Delta_{out}) \in \mathbb{F}_2^{|\Delta_{in}|+|\Delta_{out}|}$ ($|\Delta_{in}|$ is the bit length of Δ_{in}), if there exists one input-output pair (x_i, x_i') and (y_i, y_i') that satisfies the differential $\delta_{in}^{(i)} \mapsto \delta_{out}^{(i)}$ for each $\mathtt{SB}^{(i)}$ with $1 \leq i \leq l$, there will be $2^l/2 = 2^{l-1}$ choices to construct a whole state S as a starting point for the outbound phase, e.g. $S = x_1 \| x_2' \| x_3 \| \cdots \| x_l$. To indicate which state S to choose, Hosoyamda and Sasaki introduced $(l-1)$-bit $\alpha \in \mathbb{F}_2^{l-1}$ as the auxiliary variable to indicate the choice. They built a function $F(\Delta_{in}, \Delta_{out}, \alpha)$ and run Grover's algorithm on $F(\cdot)$ to find collisions. Suppose the probability of the outbound phase to find one collision is $2^{-(|\Delta_{in}|+|\Delta_{out}|)}$, in classical setting one has to traverse a domain of size $2^{|\Delta_{in}|+|\Delta_{out}|}$ for one collision. In contrast, one has to traverse $2^{|\Delta_{in}|+|\Delta_{out}|+l-1}$ in the quantum setting. Suppose the Sbox is of c-bit. The quantum time complexity will be

$$\sqrt{2^{|\Delta_{in}|+|\Delta_{out}|+l-1}} \cdot (l \cdot 2^{\frac{c}{2}}) \approx l \cdot 2^{\frac{|\Delta_{in}|+|\Delta_{out}|+l+c}{2}} \quad \text{Sbox evaluations.} \tag{3}$$

The drawback of this approach is that if there are many active Sboxes to match, the factor $l \cdot 2^{l/2}$ will have a non-negligible impact on the overall time complexity.

Our Improved Quantum Rebound Attack With QAA. Generally, as shown by Lamberger et al. [56], for any permutation Q (to be more precise, for any injective map) the expected number of solutions to $Q(\Delta_{in} \oplus x) \oplus Q(x) = \Delta_{out}$ is always 1. Hence, given a random $(\Delta_{in}, \Delta_{out})$, it is expected to find one input-output pair as the starting point. Taking the Sbox of `AES` as an example, for random given $\delta_{in}^{(i)} \mapsto \delta_{out}^{(i)}$ for $\mathtt{SB}^{(i)}$ with $1 \leq i \leq l$, $\mathtt{SB}^{(i)}(\delta_{in}^{(i)} \oplus x_i) \oplus \mathtt{SB}^{(i)}(x_i) = \delta_{out}^{(i)}$ has solution with probability about $\frac{1}{2}$, which means half of $\delta_{in}^{(i)} \mapsto \delta_{out}^{(i)}$ are impossible differentials. For a randomly given $(\Delta_{in}, \Delta_{out})$, it is possible differential with probability about $(\frac{1}{2})^l$. Hence, we have $(\frac{1}{2})^l \cdot 2^l = 1$ solution on average. Therefore, one can use the filter $(\frac{1}{2})^{-l}$ to first identify the valid $(\Delta_{in}, \Delta_{out})$, then traverse only those valid $(\Delta_{in}, \Delta_{out})$ to generate the starting points.

Concretely, we propose to use quantum amplitude amplification (QAA) algorithm for this stage. Define $g : \mathbb{F}_2^c \to \mathbb{F}_2$ to be a Boolean function such that $g(\delta_{in}, \delta_{out}, \beta = 0; x) = 1$ if and only if $\mathtt{SB}(x) \oplus \mathtt{SB}(x \oplus \delta_{in}) = \delta_{out}$ and $x \leq x \oplus \delta_{in}$, and $g(\delta_{in}, \delta_{out}, \beta = 1, x) = 1$ if and only if $\mathtt{SB}(x) \oplus \mathtt{SB}(x \oplus \delta_{in}) = \delta_{out}$ and $x > x \oplus \delta_{in}$. Define $f : \mathbb{F}_2^{|\Delta_{in}|+|\Delta_{out}|} \mapsto \mathbb{F}_2$, such that $f(\Delta_{in}, \Delta_{out}) = 1$ with $(\Delta_{in}, \Delta_{out}) \in \mathbb{F}_2^{|\Delta_{in}|+|\Delta_{out}|}$ if and only if $(\Delta_{in}, \Delta_{out})$ is a valid differential (not impossible differential). The probability of $f(\Delta_{in}, \Delta_{out}) = 1$ is 2^{-l}. The implementation of $\mathcal{U}_f$ is in Algorithm 1. The time complexity of f is $l \cdot 2^{c/2}$ in the worst case.

Algorithm 1: Implementing $\mathcal{U}_f$

Input: $|\Delta_{in}, \Delta_{out}\rangle\,|y\rangle$
Output: $|\Delta_{in}, \Delta_{out}\rangle\,|y \oplus f(\Delta_{in}, \Delta_{out})\rangle$

for $i = 1, 2, ..., l$ **do**
  Deduce the input-output difference for $\mathtt{SB}^{(i)}$, i.e., $(\delta_{in}^{(i)}, \delta_{out}^{(i)})$
  Run Grover's algorithm on $g_i(\delta_{in}^{(i)}, \delta_{in}^{(j)}, 0; \cdot) : \mathbb{F}_2^c \mapsto \mathbb{F}_2$. Let x_i be the output
  if $\mathtt{SB}^{(i)}(x \oplus \delta_{in}^{(i)}) \oplus \mathtt{SB}^{(i)}(x) \neq \delta_{out}^{(i)}$ **then**
    Return $|\Delta_{in}, \Delta_{out}\rangle\,|y\rangle$
Return $|\Delta_{in}, \Delta_{out}\rangle\,|y \oplus 1\rangle$

Define the set $X := \{(\Delta_{in}, \Delta_{out}) : (\Delta_{in}, \Delta_{out}) \in \mathbb{F}_2^{|\Delta_{in}|+|\Delta_{out}|}\}$ and $X' := \{(\Delta_{in}, \Delta_{out}) : (\Delta_{in}, \Delta_{out}) \text{ is valid differential}\} \subset X$. The size of X' is about $|X'| = 2^{|\Delta_{in}|+|\Delta_{out}|-l}$. The *Setup* oracle $\mathcal{A}$ is to run Grover's algorithm on $f(\cdot) : \mathbb{F}_2^{|\Delta_{in}|+|\Delta_{out}|} \mapsto \mathbb{F}_2$ to get the superposition $|\phi\rangle = \sum_{(\Delta_{in},\Delta_{out})\in X'} |\Delta_{in}, \Delta_{out}\rangle$. Among X', only one $(\Delta_{in}, \Delta_{out})$ can produce the good starting points and leads to the final collision. Hence, the probability of producing the good output when measuring $|\phi\rangle$ is $a = 2^{-(|\Delta_{in}|+|\Delta_{out}|-l)}$. Then, $|\mathcal{A}|_{RT} \approx 2^{l/2} \cdot l \cdot 2^{c/2} = l \cdot 2^{\frac{l+c}{2}}$.

Given $(\Delta_{in}, \Delta_{out}) \in X$, define $G : \mathbb{F}_2^{l-1} \mapsto \mathbb{F}_2$ as $G(\Delta_{in}, \Delta_{out}; \alpha)$ with $\alpha \in \mathbb{F}_2^{l-1}$. $G(\Delta_{in}, \Delta_{out}; \alpha) = 1$ if and only if $(\Delta_{in}, \Delta_{out}; \alpha)$ leads to a collision. Only one $(\Delta_{in}, \Delta_{out}) \in X'$ together with the right $\alpha \in \mathbb{F}_2^{l-1}$ lead to a collision. Hence, $G(\Delta_{in}, \Delta_{out}; \alpha) = 1$ with probability about $2^{-(|\Delta_{in}|+|\Delta_{out}|)}$. The implementation of $\mathcal{U}_G$ is given in Algorithm 2 with time complexity of $l \cdot 2^{c/2}$. Define the *Check* operator $\mathcal{P} : \mathbb{F}_2^{|\Delta_{in}|+|\Delta_{out}|} \mapsto \mathbb{F}_2$ as $\mathcal{P}(\Delta_{in}, \Delta_{out})$ with $(\Delta_{in}, \Delta_{out}) \in X'$. $\mathcal{P}(\Delta_{in}, \Delta_{out}) = 1$ if and only if $(\Delta_{in}, \Delta_{out})$ leads to a collision. The implementation of $\mathcal{U}_\mathcal{P}$ is given in Algorithm 3 with time complexity $|\mathcal{U}_\mathcal{P}|_{RT} = 2^{(l-1)/2} \cdot l \cdot 2^{c/2} = l \cdot 2^{\frac{l+c-1}{2}}$. $\mathcal{P}(\Delta_{in}, \Delta_{out}) = 1$ with probability of $a = \frac{1}{|X'|} = 2^{-(|\Delta_{in}|+|\Delta_{out}|-l)}$. Hence, according to Equation 2, the total time complexity is

$$\begin{aligned}\sqrt{1/a} \cdot (|\mathcal{A}|_{RT} + |\mathcal{U}_\mathcal{P}|_{RT}) &= \sqrt{2^{|\Delta_{in}|+|\Delta_{out}|-l}} \cdot (l \cdot 2^{\frac{l+c}{2}} + l \cdot 2^{\frac{l+c-1}{2}}) \\ &\approx l \cdot 2^{\frac{|\Delta_{in}|+|\Delta_{out}|+c}{2}} \quad \text{Sbox evaluations.}\end{aligned} \tag{4}$$

Compared with [39] of complexity in Equation 3, our approach leads to a reduction of time complexity by a factor of roughly $2^{l/2}$, so that the number of active Sboxes to be matched only has a linear rather than exponential impact.

A Case Study of `AES`. At ToSC 2019, Bonnetain, Naya-Plasencia, and Schrottenloher [14] proposed the quantum circuit of `DDT` for `AES`'s Sbox.

Property 2 ([14]). There exists a quantum unitary for the `DDT` of `AES` Sbox that, for a given $(\delta_{in}, \delta_{out})$, finds a solution x conforming the difference pair and outputs (x, OK) if such an x exists, and $(0, none)$ otherwise. The circuit comes with

Algorithm 2: Implementing $\mathcal{U}_G$

Input: $|\Delta_{in}, \Delta_{out}; \alpha\rangle\,|y\rangle$, $\alpha = (\alpha_1, \alpha_2, ..., \alpha_{l-1}) \in \mathbb{F}_2^{l-1}$
Output: $|\Delta_{in}, \Delta_{out}; \alpha\rangle\,|y \oplus G(\Delta_{in}, \Delta_{out}; \alpha)\rangle$

for $i = 1, 2, ..., l-1$ **do**
  Deduce the input-output difference for $\texttt{SB}^{(i)}$, i.e., $(\delta_{in}^{(i)}, \delta_{out}^{(i)})$
  Run Grover's algorithm on $g_i(\delta_{in}^{(i)}, \delta_{in}^{(j)}, \alpha_i; \cdot) : \mathbb{F}_2^c \mapsto \mathbb{F}_2$. Let x_i be the output
Run Grover's algorithm on $g_l(\delta_{in}^{(l)}, \delta_{in}^{(l)}, 0; \cdot) : \mathbb{F}_2^c \mapsto \mathbb{F}_2$. Let x_l be the output
Let $x_1 \| x_2, ..., \| x_l$ be the starting point
if $x_1 \| x_2, ..., \| x_l$ *leads to a collision* **then**
  return $|\Delta_{in}, \Delta_{out}; \alpha\rangle\,|y \oplus 1\rangle$
else
  return $|\Delta_{in}, \Delta_{out}; \alpha\rangle\,|y\rangle$

Algorithm 3: Implementing $\mathcal{U}_{\mathcal{P}}$: Checking if $(\Delta_{in}, \Delta_{out})$ leads to a collision

Input: $|\Delta_{in}, \Delta_{out}\rangle\,|y\rangle$
Output: $|\Delta_{in}, \Delta_{out}\rangle\,|y \oplus \mathcal{P}(\Delta_{in}, \Delta_{out})\rangle$

Run Grover's algorithm on $G(\Delta_{in}, \Delta_{out}; \cdot) : \mathbb{F}_2^{l-1} \mapsto \mathbb{F}_2$. Let $\alpha \in \mathbb{F}_2^{l-1}$ be the output
/* If $(\Delta_{in}, \Delta_{out})$ leads to a collision, which happens with probability of $\frac{1}{|X'|} = 2^{-(|\Delta_{in}|+|\Delta_{out}|-l)}$, the right α will output in Step 1; Else, a random α will output */
if $G(\Delta_{in}, \Delta_{out}; \alpha) = 1$ **then**
  return $|\Delta_{in}, \Delta_{out}\rangle\,|y \oplus 1\rangle$
else
  return $|\Delta_{in}, \Delta_{out}\rangle\,|y\rangle$

22 ancilla qubits and a time complexity equivalent with 2 SBox computations. If the goal is only to decide whether a solution exists but not to find out it explicitly, the cost drops to 1 SBox computation and 15 ancilla qubits.

According to Property 2, $|\mathcal{A}|_{RT} = 2^{l/2} \cdot l \cdot 1 = l \cdot 2^{l/2}$ and $|\mathcal{U}_{\mathcal{P}}|_{RT} = 2^{(l-1)/2} \cdot l \cdot 2 = l \cdot 2^{l+1}$. Then, Equation 4 becomes:

$$\begin{aligned}\sqrt{|X'|} \cdot (|\mathcal{A}|_{RT} + |\mathcal{U}_{\mathcal{P}}|_{RT}) &= \sqrt{2^{|\Delta_{in}|+|\Delta_{out}|-l}} \cdot (l \cdot 2^{\frac{l}{2}} + l \cdot 2^{\frac{l+1}{2}}) \\ &\approx l \cdot 2^{\frac{|\Delta_{in}|+|\Delta_{out}|}{2}} \quad \text{Sbox evaluations.}\end{aligned} \tag{5}$$

B Detailed Quantum Collision Attack on 8-round AES-128

As shown in Figure 15, focusing on the first Super-Sbox $\texttt{SSB}^{(0)}$ marked by red box, for the fixed chosen difference Δx_3, Δy_4 and a given k_3, we apply

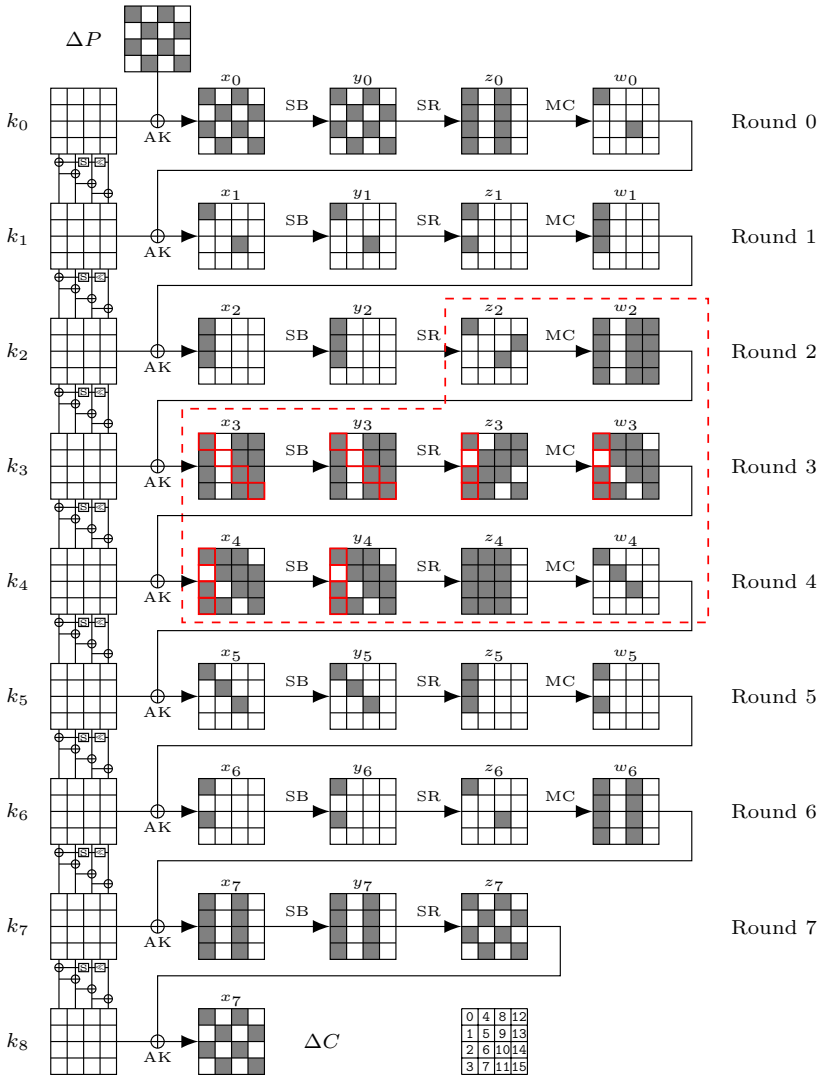


Fig. 15: Quantum collision attack on 8-round AES-128

QAA algorithm to compute the conforming pair for the Super-Sbox $\texttt{SSB}^{(0)}$. Define $g_0(\Delta x_3^{(0)}, \Delta y_4^{(0)}; \Delta w_3[0,2]) : \mathbb{F}_2^{8+8} \mapsto \mathbb{F}_2$, where $(\Delta x_3^{(0)}, \Delta y_4^{(0)})$ are marked by red box and involved in $\texttt{SSB}^{(0)}$. It marks the valid $\Delta w_3[0,2]$ that makes $\Delta x_3^{(0)} \to \Delta w_3[0,2] \to y_4^{(0)}$ a possible differential. The implementation of $\mathcal{U}_{g_0}$ is given in Algorithm 4 with time complexity of about 6 Sbox evaluations. The probability of $g_0(\Delta x_3^{(0)}, \Delta y_4^{(0)}; \Delta w_3[0,2]) = 1$ is about 2^{-6}. Define set $S_{(\Delta x_3, \Delta y_4)} := \{\Delta w_3[0,2] : \Delta x_3^{(0)} \to \Delta w_3[0,2] \to y_4^{(0)}$ is a possible differential$\}$. The size $|S_{(\Delta x_3, \Delta y_4)}| = 2^{16-6} = 2^{10}$. The setup oracle $\mathcal{A}_0$ is to run Grover's

algorithm on $g_0(\Delta x_3^{(0)}, \Delta y_4^{(0)}; \cdot) : \mathbb{F}_2^{16} \mapsto \mathbb{F}_2$ to get the superposition $|\phi\rangle = \sum_{\Delta w_3[0,2] \in S_{(\Delta x_3, \Delta y_4)}} |\Delta w_3[0,2]\rangle$ with time complexity of $\frac{\pi}{4} \cdot \sqrt{2^6} \cdot 6 = 2^{5.238}$ Sbox evaluations.

Algorithm 4: Quantum Implementation of $\mathcal{U}_{g_0}$

Input: $|\Delta x_3^{(0)}, \Delta y_4^{(0)}; \Delta w_3[0,2]\rangle\,|y\rangle$
Output: $|\Delta x_3^{(0)}, \Delta y_4^{(0)}; \Delta w_3[0,2]\rangle\,|y \oplus g_0(\Delta x_3^{(0)}, \Delta y_4^{(0)}; \Delta w_3[0,2])\rangle$

Deduce $\Delta z_3[0,2,3]$, $\Delta w_3[3]$ with known differences $\Delta w_3[0,2]$ and $\Delta z_3[1] = 0, \Delta w_3[1] = 0$ according to Property 1
For $(\delta_{in}, \delta_{out})$ of the Sbox, apply quantum circuit of `DDT` to check if $(\delta_{in}, \delta_{out})$ is a valid differential. Define $\mathcal{U}_{\texttt{DDT}}(\delta_{in}, \delta_{out}) = 1$ if and only if $(\delta_{in}, \delta_{out})$ is a valid differential
`/* According to Property 2, the cost equals to 1 Sbox computation with 15 ancilla qubits */`
if $\mathcal{U}_{\texttt{DDT}}(\Delta x_3[0], \Delta y_3[0]) = 0$ **then**
    return $|\Delta x_3^{(0)}, \Delta y_4^{(0)}; \Delta w_3[0,2]\rangle\,|y\rangle$
if $\mathcal{U}_{\texttt{DDT}}(\Delta x_3[10], \Delta y_3[10]) = 0$ **then**
    return $|\Delta x_3^{(0)}, \Delta y_4^{(0)}; \Delta w_3[0,2]\rangle\,|y\rangle$
⋮
if $\mathcal{U}_{\texttt{DDT}}(\Delta x_4[3], \Delta y_4[3]) = 0$ **then**
    return $|\Delta x_3^{(0)}, \Delta y_4^{(0)}; \Delta w_3[0,2]\rangle\,|y\rangle$
`/* Totally, six input-output differences are checked */`
return $|\Delta x_3^{(0)}, \Delta y_4^{(0)}; \Delta w_3[0,2]\rangle\,|y \oplus 1\rangle$

For a given key $k_4^{(0)} \in \mathbb{F}_2^{32}$ involved in $\texttt{SSB}^{(0)}$, define the projector $\mathcal{P}_0 : \mathbb{F}_2^{16} \mapsto \mathbb{F}_2$, where $\mathcal{P}_0(k_4^{(0)}, \Delta x_3, \Delta y_4; \Delta w_3[0,2]) = 1$ if and only if there is a pair conforming $\Delta x_3 \mapsto \Delta w_3[0,2] \mapsto \Delta y_4$ under $k_4^{(0)}$. Before implement $\mathcal{U}_{\mathcal{P}_0}$, we define a new $\mathcal{U}_{f_0} : \mathbb{F}_2^6 \mapsto \mathbb{F}_2$, where $\mathcal{U}_{f_0}(k_4^{(0)}, \Delta x_3, \Delta y_4, \Delta w_3[0,2]; \alpha) = 1$ if and only if $(k_4^{(0)}, \Delta x_3, \Delta y_4, \Delta w_3[0,2]; \alpha)$ outputs a valid pair for the differential with $\alpha \in \mathbb{F}_2^6$. The implementation of $\mathcal{U}_{f_0}$ is given in Algorithm 5 with time complexity of 15 Sbox evalutions. The $\mathcal{U}_{\mathcal{P}_0}$ is implemented in Algorithm 6 with time complexity of about $\frac{\pi}{4} \cdot \sqrt{2^6} \cdot 15 = 2^{6.56}$ Sbox evalutions.

At last, given $(k_4^{(0)}, \Delta x_3, \Delta y_4)$ for $\texttt{SSB}^{(0)}$, by applying QAA algorithm (defined as $\mathcal{U}_{\texttt{SSB}^{(0)}}$), we find the right $\Delta w_3[0,2] = \mathcal{U}_{\texttt{SSB}^{(0)}}(k_4^{(0)}, \Delta x_3, \Delta y_4)$ with time complexity $\frac{\pi}{4} \cdot \sqrt{|S_{(\Delta x_3, \Delta y_4)}|} \cdot (2^{5.238} + 2^{6.56}) = 2^{11.7}$ Sbox evaluations according to Equation 2.

Similarly, we define $\mathcal{U}_{\texttt{SSB}^{(1)}}$, $\mathcal{U}_{\texttt{SSB}^{(2)}}$, $\mathcal{U}_{\texttt{SSB}^{(3)}}$ to compute $\Delta w_3[4,5]$, $\Delta w_3[8,9]$, $\Delta w_3[13,14]$ in parallel.

We use two blocks to build the collision as shown in Figure 11. We define $G(m_0, \Delta z_2, \Delta w_4, \beta)$, where $\Delta z_2 \in \mathbb{F}_2^{24}$, $\Delta w_4 \in \mathbb{F}_2^{24}$, and $\beta = (\beta_0, \beta_1, \beta_2) \in \mathbb{F}_2^3$. $G(m_0, \Delta z_2, \Delta w_4, \beta) = 1$ if and only if $(m_0, \Delta z_2, \Delta w_4, \beta)$ leads to a collision.

Algorithm 5: Quantum Implementation of $\mathcal{U}_{f_0}$

Input: $|k_4^{(0)}, \Delta x_3, \Delta y_4, \Delta w_3[0,2]; \alpha\rangle$, $\alpha = (\alpha_0, \alpha_1, ..., \alpha_5) \in \mathbb{F}_2^6$
Output: $|k_4^{(0)}, \Delta x_3, \Delta y_4, \Delta w_3[0,2]; \alpha\rangle\,|y \oplus f_0(k_4^{(0)}, \Delta x_3, \Delta y_4, \Delta w_3[0,2]; \alpha\rangle$

1 Deduce $\Delta z_3[0,2,3]$, $\Delta w_3[3]$ with known differences $\Delta w_3[0,2]$ and $\Delta z_3[1] = 0, \Delta w_3[1] = 0$ according to Property 1.
2 For $(\delta_{in}, \delta_{out})$ of the Sbox, apply quantum circuit of DDT to output the conforming value. Let $\mathcal{U}'_{\texttt{DDT}}(\delta_{in}, \delta_{out}) = x$, where $\texttt{SB}(x \oplus \delta_{in}) \oplus \texttt{SB}(x) = \delta_{out}$
3 /* According to Property 2, the cost equals to 2 Sbox computation with 22 ancilla qubits */
4 **if** $\alpha_0 = 1$ **then**
5 $\quad x_3[0] = \mathcal{U}'_{\texttt{DDT}}(\Delta x_3[0], \Delta y_3[0])$
6 **else**
7 $\quad x_3[0] = \mathcal{U}'_{\texttt{DDT}}(\Delta x_3[0], \Delta y_3[0]) \oplus \Delta x_3[0]$
8 **if** $\alpha_1 = 1$ **then**
9 $\quad x_3[10] = \mathcal{U}'_{\texttt{DDT}}(\Delta x_3[10], \Delta y_3[10])$
10 **else**
11 $\quad x_3[0] = \mathcal{U}'_{\texttt{DDT}}(\Delta x_3[10], \Delta y_3[10]) \oplus \Delta x_3[10]$
12 $\vdots$
13 **if** $\alpha_5 = 1$ **then**
14 $\quad x_4[3] = \mathcal{U}'_{\texttt{DDT}}(\Delta x_4[3], \Delta y_4[3])$
15 **else**
16 $\quad x_4[3] = \mathcal{U}'_{\texttt{DDT}}(\Delta x_4[3], \Delta y_4[3]) \oplus \Delta x_4[3]$
17 Compute $w_3[0,2,3] = k_4[0,2,3] \oplus x_4[0,2,3]$ and $z_3[0,2,3] = \texttt{SB}(x_3[0,10,15])$
18 **if** $w_3[0,2,3] = \texttt{MC}(z_3[0,2,3])$/* According to Property 1, it holds with probability of 2^{-16} */
19 **then**
20 $\quad |k_4^{(0)}, \Delta x_3, \Delta y_4, \Delta w_3[0,2]; \alpha\rangle\,|y \oplus 1\rangle$
21 **else**
22 $\quad |k_4^{(0)}, \Delta x_3, \Delta y_4, \Delta w_3[0,2]; \alpha\rangle\,|y\rangle$

The complexity to implement $\mathcal{U}_G$ in Algorithm 7 is bounded by Line 4 to 4, which is about $2^{11.7} \times 4 = 2^{13.7}$ Sbox evaluations. Since the probability of the outbound phase is 2^{-96}, together with the three auxiliary qubits β, $G(m_0, \Delta z_2, \Delta w_4, \beta) = 1$ with probability of 2^{-99}. We build U_F in Algorithm 8 to finally obtain the collision, whose time complexity is bounded by Line 1 with $\frac{\pi}{4} \cdot \sqrt{2^{99}} \cdot 2^{13.7} = 2^{62.85}$ Sbox evaluations. There are 160 Sboxes in an 8-round AES. Hence, the total time complexity to get the collision (m_0, m_1, m'_1) is about $2^{62.85}/160 = 2^{55.53}$ 8-round AES computations.

Algorithm 6: Quantum Implementation of $\mathcal{U}_{\mathcal{P}_0}$

Input: $|k_4^{(0)}, \Delta x_3, \Delta y_4; \Delta w_3[0,2]\rangle$
Output: $|k_4^{(0)}, \Delta x_3, \Delta y_4; \Delta w_3[0,2]\rangle\,|y \oplus f_0(k_4^{(0)}, \Delta x_3, \Delta y_4; \Delta w_3[0,2]\rangle$

Run Grover's algorithm on $\mathcal{U}_{f_0}(k_4^{(0)}, \Delta x_3, \Delta y_4, \Delta w_3[0,2]; \cdot) : \mathbb{F}_2^6 \mapsto \mathbb{F}_2$, let α as output
Run Line to of Algorithm 5 with α to output $w_3[0,2,3]$ and $z_3[0,2,3]$
if $w_3[0,2,3] = \texttt{MC}(z_3[0,2,3])$ **then**
  $|k_4^{(0)}, \Delta x_3, \Delta y_4; \Delta w_3[0,2]\rangle\,|y \oplus 1\rangle$
else
  $|k_4^{(0)}, \Delta x_3, \Delta y_4; \Delta w_3[0,2]\rangle\,|y\rangle$

Algorithm 7: Quantum Implementation of $\mathcal{U}_G$

Input: $|m_0, \Delta z_2, \Delta w_4, \beta\rangle$, $\beta = (\beta_0, \beta_1, \beta_2) \in \mathbb{F}_2^3$
Output: $|m_0, \Delta z_2, \Delta w_4, \beta\rangle\,|y \oplus G(m_0, \Delta z_2, \Delta w_4, \beta)\rangle$

Comput k_4 for the second block from m_0
Deduce Δx_3, Δy_4 from Δz_2 and Δy_4
Compute $x_3[0,10,15]$ by accessing the quantum circuit of DDT with input-output differences, then $z_3[0,2,3]$ are known
Compute $\Delta w_3[0,2] = \mathcal{U}_{\texttt{SSB}^{(0)}}(k_4^{(0)}, \Delta x_3, \Delta y_4)$
Compute $\Delta w_3[4,5] = \mathcal{U}_{\texttt{SSB}^{(1)}}(k_4^{(1)}, \Delta x_3, \Delta y_4)$
Compute $\Delta w_3[8,9] = \mathcal{U}_{\texttt{SSB}^{(2)}}(k_4^{(2)}, \Delta x_3, \Delta y_4)$
Compute $\Delta w_3[13,14] = \mathcal{U}_{\texttt{SSB}^{(3)}}(k_4^{(3)}, \Delta x_3, \Delta y_4)$
Run $\mathcal{U}_{\mathcal{P}_0}$ with $\Delta w_3[0,2]$, let $w_3[0,2,3]$ and $z_3[0,2,3]$. Compute $x_4[0,1,2,3]$ from $w_3[0,2,3]$, $z_3[0,2,3]$ and k_4. Let $x_4[0,1,2,3]$ as starting point if $\beta_0 = 1$, else $x_4[0,1,2,3] \oplus \Delta x_4[0,1,2,3]$ as starting point
Run $\mathcal{U}_{\mathcal{P}_1}$ with $\Delta w_3[4,5]$, let $w_3[4,5,7]$ and $z_3[5,6,7]$. Compute $x_4[4,5,6,7]$ from $w_3[4,5,7]$ and $z_3[5,6,7]$ and k_4. Let $x_4[4,5,6,7]$ be the starting point if $\beta_1 = 1$, else $x_4[4,5,6,7] \oplus \Delta x_4[4,5,6,7]$ be starting point
Run $\mathcal{U}_{\mathcal{P}_2}$ with $\Delta w_3[8,9]$, let $w_3[8,9,10]$ and $z_3[8,9,10]$. Compute $x_4[8,9,10,11]$ from $w_3[8,9,10]$ and $z_3[8,9,10]$ and k_4. Let $x_4[8,9,10,11]$ be the starting point if $\beta_2 = 1$, else $x_4[8,9,10,11] \oplus \Delta x_4[8,9,10,11]$ be starting point
Run $\mathcal{U}_{\mathcal{P}_3}$ with $\Delta w_3[13,14]$, let $w_3[13,14,15]$ and $z_3[12,13,15]$. Compute $x_4[12,13,14,15]$ from $w_3[13,14,15]$ and $z_3[12,13,15]$ and k_4. Let $x_4[12,13,14,15]$ be the starting point
if *Check if the starting point lead to a collision* **then**
  return $|m_0, \Delta z_2, \Delta w_4, \beta\rangle\,|y \oplus 1\rangle$
else
  return $|m_0, \Delta z_2, \Delta w_4, \beta\rangle\,|y\rangle$

Algorithm 8: Quantum Implementation of $\mathcal{U}_F$

Output: m_0, m_1, m_1'

1 Run Grover's algorithm on $G(\cdot) : \mathbb{F}_2^{128+24+24+3} \mapsto \mathbb{F}_2$ and let $(m_0, \Delta z_2, \Delta w_4, \beta)$ be the output
2 Run Line 1 to 11 of Algorithm 7 to get the x_4
3 Compute $x_4' = x_4 \oplus \Delta x_4$
4 Compute m_1 from x_4 and m_0
5 Compute m_1' from x_4' and m_0
6 return m_0, m_1, m_1'

C Practical Semi-Free-Start Collision 6-round AES-128

Table 4 presents an example semi-free-start collision on 6-round AES. The code to generate the collision is given at https://www.dropbox.com/s/2n4d9zwufkpigqt/Find_inbound.7z

D Quantum 7-Round Semi-Free-Start Collision Attack on Saturnin-Hash

We derive a semi-free-start collision attack on 7-round Saturnin by using the new truncated differential trail shown in Figure 16. The inbound phase is separated into 2 phases, while the outbound phase happens with probability 2^{-96} for the MC cancellation at round 5. Combine with 4-byte cancellation feed-forward operator, we obtain an collision attack with probability 2^{-160}.

Inbound phase details. Given $(\Delta y_1, \Delta w_3, \Delta w_4)$ as 3 prefixed differences of the inbound phase. Our following details describe an efficient way to generate the data pairs and a key conforming the inbound path.

1. Compute $\Delta x_2 = \mathtt{MC}(\Delta y_1)$, $\Delta y_3 = \mathtt{MC}^{-1}(\Delta w_3)$, $\Delta y_4 = \mathtt{MR}^{-1}(\Delta w_4)$, and $\Delta x_4 = \Delta w_3$.
2. Deduce the active bytes value of (x_4, x_4') by assessing to the DDT.
3. Assign random values to first column of K, i.e. byte k_0, k_4, k_8, k_{12}, then compute $x_3[0,4,8,12] = \mathtt{SB}^{-1}(\mathtt{MC}^{-1}(x_4[0,4,8,12] \oplus k_{\{0,4,8,12\}}))$ and the corresponding $\Delta x_3[0,4,8,12]$.
4. From $\Delta w_2[0,4,8,12] = \Delta x_3[0,4,8,12]$ and Property 1, we know all the differences of Δy_2 and Δw_2. The active byte values of x_2 and y_2 are determined as well.
5. Since $\Delta x_3 = \Delta w_2$, we assess to the DDT to obtain the remaining active byte values of x_3 and y_3. We also obtain the remaining byte values at second and third column of w_3.
6. Connect w_3 and x_4 by choosing $k_{\{1,5,9,13,2,6,10,14\}}$, then the first and second columns of K' are known as well.

Plaintext			
P	13622301 f4ad7096 c7cea69e e26646e5	K_{-1}	6aa0c09f 92854210 9411daed 8db5f736
P'	eb622301 f4ad7096 c7cedd9e e26646e5		
After AK			
X_0	79c2e39e 66283286 53df7c73 6fd3b1d3		
X_0'	81c2e39e 66283286 53df0773 6fd3b1d3		
After SB&SR			
Z_0	b6341066 339ec80b ed661144 a825238f		
Z_0'	0c34c566 339ec80b ed661144 a825238f		
After MC			
W_0	5d880829 1c5c3b15 3e5665d3 880841e0	K_0	bec8c5c2 2c4d87d2 b85c5d3f 35e9aa09
W_0'	e7560329 1c5c3b15 3e5665d3 880841e0		
After AK			
X_1	e340cdeb 3011bcc7 860a38ec bde1ebe9		
X_1'	599ec6eb 3011bcc7 860a38ec bde1ebe9		
After SB&SR			
Z_1	1182071e 0467e9e9 44f8bdc6 7a0965ce		
Z_1'	cb82071e 0467e9e9 44f8b4c6 7a0b65ce		
After MC			
W_1	a619bf8a a1038a4b e0b58c1e 4409f065	K_1	a264c454 8e294386 36751eb9 039cb4b0
W_1'	09c365ff a1038a4b e9ae9e17 420df267		
After AK			
X_2	047d7bde 2f2ac9cd d6c092a7 479544d5		
X_2'	aba7a1ab 2f2ac9cd dfdb80ae 419146d7		
After SB&SR			
Z_2	f2e54f03 15ba1b1d f62a21bd a0ffdd5c		
Z_2'	62e5cd0e 15b95a62 9e8132bd 835cdde4		
After MC			
W_2	87f18ca1 f94abea4 157c426b c0651a61	K_2	78e9232f f6c060a9 c0b57e10 c329caa0
W_2'	33f11492 c2f0be18 306ca76b c0a349cc		
After AK			
X_3	ff18af8e 0f8ade0d d5c93c7b 034cd0c1		
X_3'	4b1837bd 3430deb1 f0d9d97b 038a836c		
After SB&SR			
Z_3	167eeb78 76dd7019 032979d7 7bad1d21		
Z_3'	b3043550 1835ec7a 8c7e9ac8 7bad1d21		
After MC			
W_3	3db42d5f f95e6005 d30dbae0 263c8f7f	K_3	d59dc301 235da3a8 e3e8ddb8 20c11718
W_3'	14b42d5f f9276005 d30d9ee0 263c8f7f		
After AK			
X_4	e829ee5e da03c3ad 30e56758 06fd9867		
X_4'	c129ee5e da7ac3ad 30e54358 06fd9867		
After SB&SR			
Z_4	9b7b8585 57d94658 04542895 6fa52e6a		
Z_4'	78da1a85 57d94658 04542895 6fa52e6a		
After MC			
W_4	a07c6559 c06cead6 4941a441 6e2628ee	K_4	bd6d6eb6 9e30cd1e 7dd810a6 5d1907be
W_4'	1a7c0259 c06cead6 4941a441 6e2628ee		
After AK			
X_5	1d110bef 5e5c27c8 3499b4e7 333f2f50		
X_5'	a7116cef 5e5c27c8 3499b4e7 333f2f50		
After SB&SR			
Z_5	a44a8d53 58ee15df 18752be8 c382cc94	K_5	49a8c0fa d7980de4 aa401d42 f7591afc
Z_5'	5c4a8d53 58ee15df 187550e8 c382cc94		
Ciphertext			
C	ede24da9 8f76183b b23536aa 34dbd668		
C'	15e24da9 8f76183b b2354daa 34dbd668		
Hash output after feedback			
H	fe806ea8 7bdb68ad 75fb9034 d6bd90cd		
H'	fe806ea8 7bdb68ad 75fb9034 d6bd90cd		

Table 4: Pair found by semi-free-start collision attack on 6-round AES-128

7. Compute the first two columns $w_2[0,4,8,12,1,5,9,13] = x_3[0,4,8,12,1,5,9,13] \oplus k_{\{5,9,13,1,6,10,14,2\}}$.
8. With known values for the two active columns of y_2 and two deduced columns of w_2 in Step 7, all the unknown values for states y_2 and w_2 are known by Property 1. Then, we deduce the last unknown key cells: $k_{\{7,11,15,3\}} = x_3[2,6,10,14] \oplus w_2[2,6,10,14]$.

The probability of the outbound phase is 2^{-160}. According to Equation 4, $l = 8 + 12 + 12 = 32$, the time complexity is $32 \cdot 2^{160/2+16/2}/(7 \times 16) = 2^{86}$ 7-round `Saturnin-Hash`.

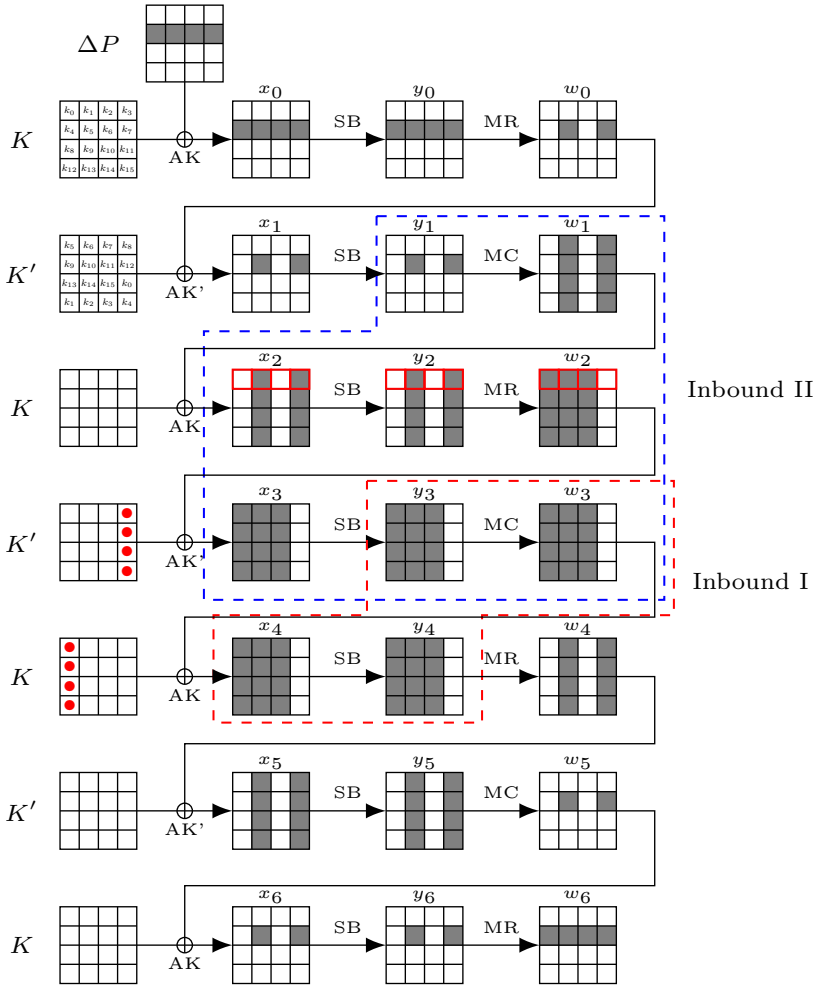


Fig. 16: Semi-free-start collision attack on 7-round `Saturnin`

E The 10-round Differential Characteristic of Saturinin-Hash

The 10-round rebound-attack trail on Saturnin-Hash is given in Figure 17. The probability of the outbound phase is $2^{-12-59.8-16-59.8-64} = 2^{-211.6}$. Together with a filter of 2^{-32} in the inbound phase, the total probability is $2^{-211.6-32} = 2^{-243.6}$.

F Figures on the Quantum Collision Attack on SKINNY-128-384

The 21-round SKINNY-128-384 using the differential characteristic shown in Figure 18 in quantum setting.

The classical free-start collision attack and a 5-round inbound data pair of 19-round SKINNY-128-384 hash mode which are shown in Figure 19 and Figure 20 respectively. The source code to generate the data of Figure 20 is at https://www.dropbox.com/s/2n4d9zwufkpigqt/Find_inbound.7z

G Quantum Collision Attacks on Reduced Grøstl

In this section, we describe an example technique to bridge the differences with the degree of freedom from the internal states. At FSE 2011, Jean et al. [45] considered the inbound phase covering three rounds on Grøstl-256 and Grøstl-512. Given input and output differences, Jean et al. built several tables for Super-Sboxes, and connect them with a guess-and-determine approach. At EUROCRYPT 2020, Hosoyamada and Sasaki [39] proposed a memoryless method to solve the 3-round inbound phase for the case like Grøstl-256 and Whirlpool, where the Sboxes in the 3-round inbound phase are all active. For the case Grøstl-512, the 3-round inbound phase is non-full-active, which means we may have many degrees of freedom from the internal states to link the differences.

Denote 'V' as the byte with known value and difference, and 'D' as the byte with known difference but no value. Denote $S[\mathrm{V}]$ as the bytes of state S marked by 'V', and $S[\mathrm{D}]$ as the bytes marked by 'D'. We explain our technique with the problem shown in Figure 21 (a) as a start: given input values $(S_1[0], S'_1[0])$ and $(S_1[1], S'_1[1])$, output differences $\Delta S_3[0]$, $\Delta S_3[2]$ (marked by blue) and $\Delta S_3[1,3] = 0$, the goal is to find a pair that conforms to the truncated differential. Formally, the pair finding is equivalent with solving the following nonlinear equation system:

$$\begin{cases} \mathtt{SB}(2S_1[0] \oplus 3S_1[1] \oplus S_1[2] \oplus S_1[3]) \oplus \mathtt{SB}(2S'_1[0] \oplus 3S'_1[1] \oplus S'_1[2] \oplus S'_1[3]) = \Delta S_3[0], \\ \mathtt{SB}(S_1[0] \oplus 2S_1[1] \oplus S_1[2] \oplus 3S_1[3]) \oplus \mathtt{SB}(S'_1[0] \oplus 2S'_1[1] \oplus 3S'_1[2] \oplus S'_1[3]) = 0, \\ \mathtt{SB}(S_1[0] \oplus S_1[1] \oplus 2S_1[2] \oplus 3S_1[3]) \oplus \mathtt{SB}(S'_1[0] \oplus S'_1[1] \oplus 2S'_1[2] \oplus 3S'_1[3]) = \Delta S_3[2], \\ \mathtt{SB}(3S_1[0] \oplus S_1[1] \oplus S_1[2] \oplus 2S_1[3]) \oplus \mathtt{SB}(3S'_1[0] \oplus S'_1[1] \oplus 2S'_1[2] \oplus S'_1[3]) = 0, \end{cases} \tag{6}$$

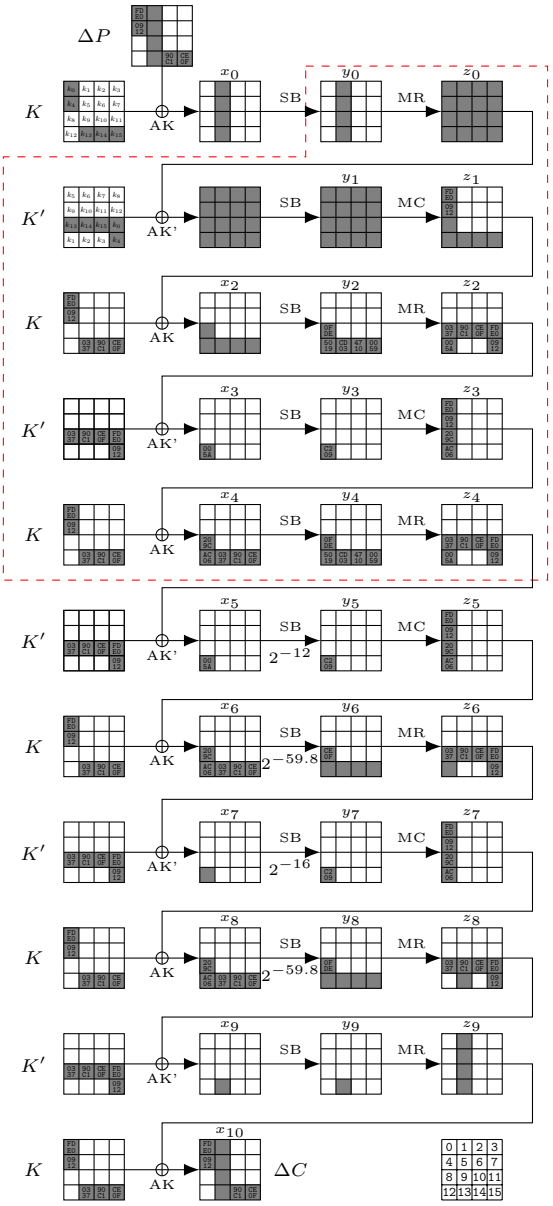


Fig. 17: Free-start collision attack on 10-round `Saturnin-Hash`

where the blue bytes are known, i.e., the equation system has 4 variables (they act as the degrees of freedom) and four equations. Hence, it is expected to have one solution on average. However, solving the nonlinear system directly is not easy. The trivial way is to traverse the 4-byte $(S_1[2], S_1[3], S_1'[2], S_1'[3])$ and check

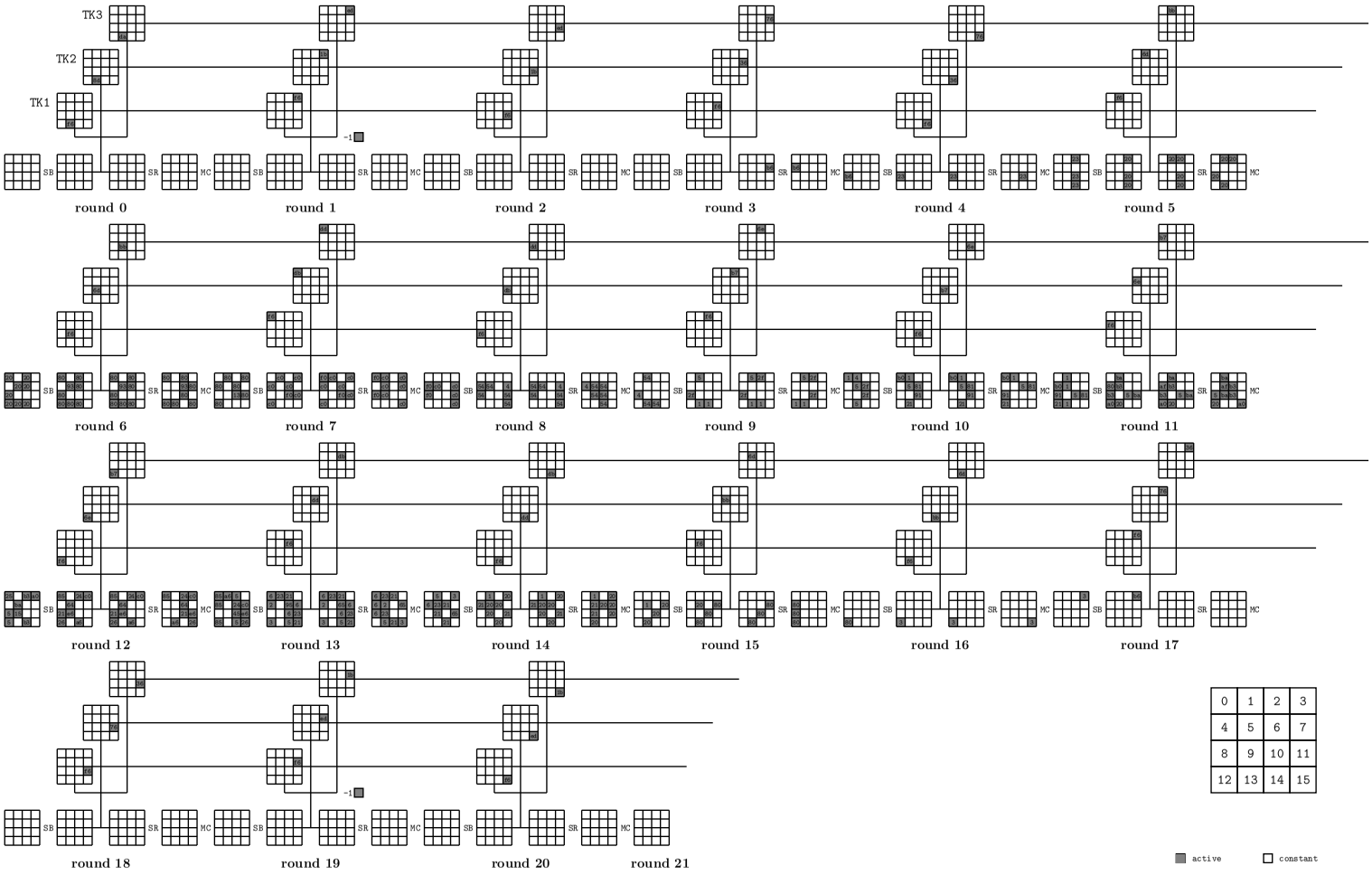


Fig. 18: Free-start collision attack on 21-round SKINNY-128-384

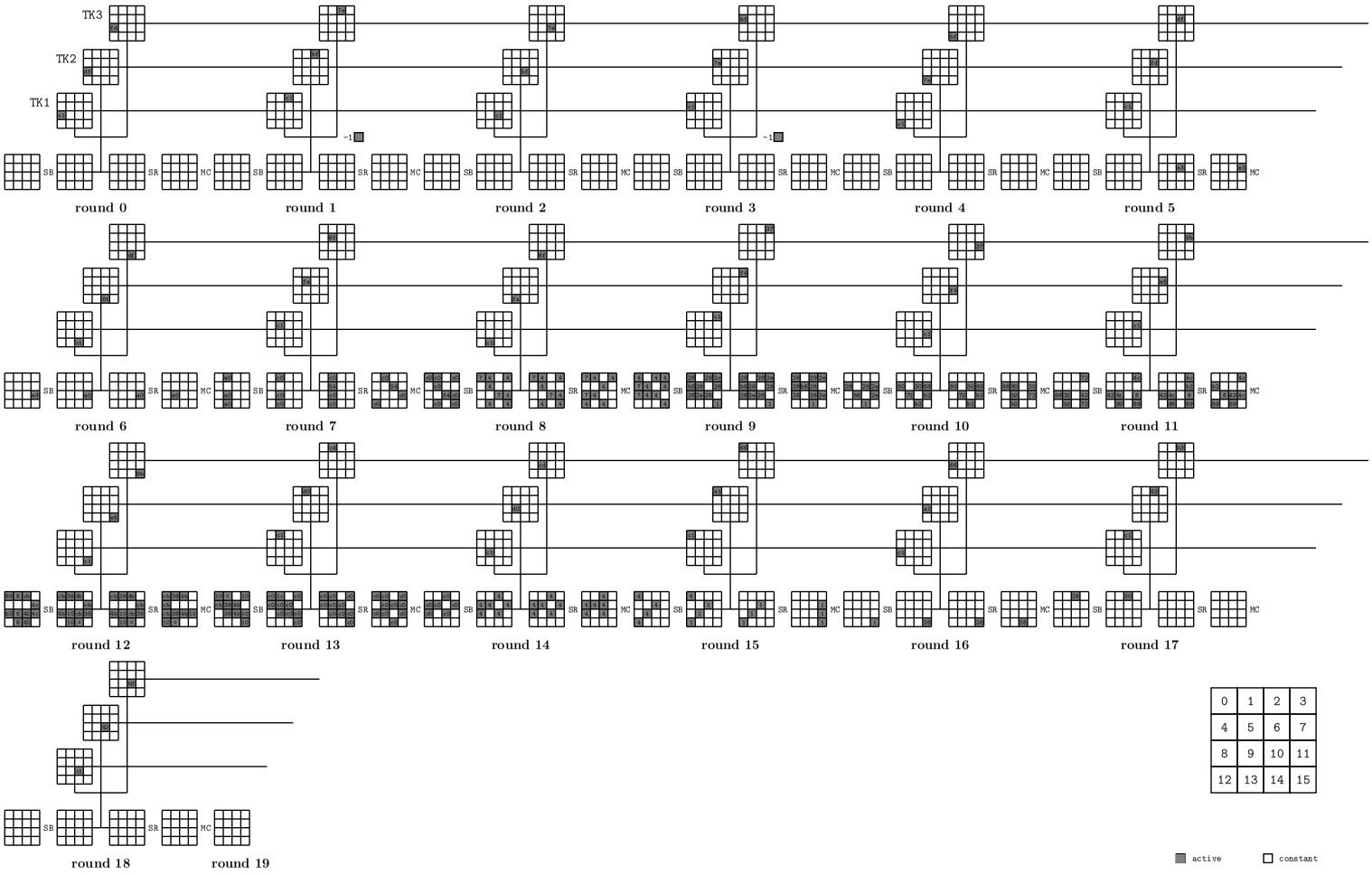


Fig. 19: Free-start collision attack on 19-round SKINNY-128-384

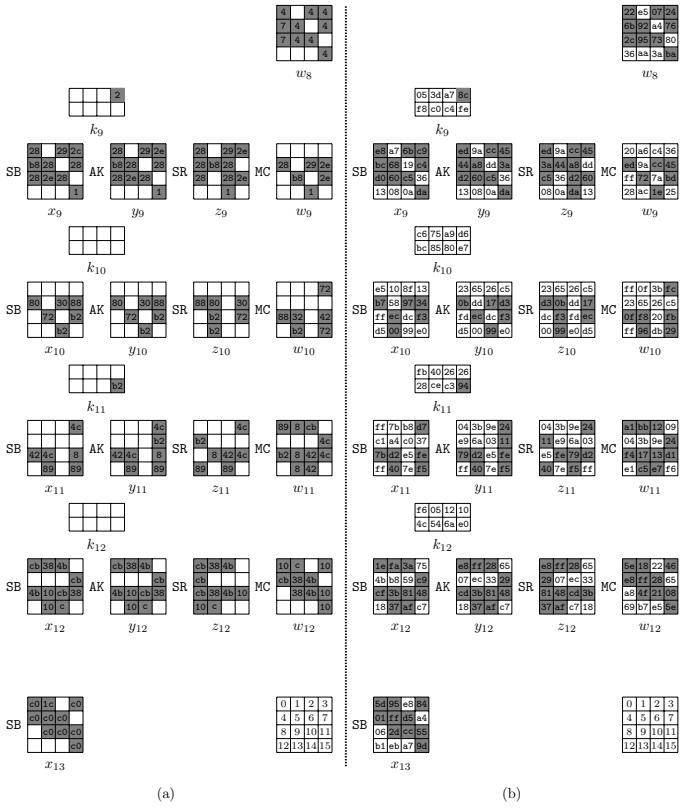


Fig. 20: 5-round inbound phase of 19-round `SKINNY-128-384`: (a) Differences, (b) Values

if the equation system holds, which comes with a time complexity of 2^{32}. Here, we give an efficient method to deduce the red bytes from blue bytes, and find the solution with time complexity 1:

1. With difference $\Delta S_1[0,1]$ and 2 inactive bytes in S_2, all other byte differences in both S_1 and S_2 can be deduced according to Property 1,
2. With $\Delta S_2[0]$, and $\Delta S_3[0]$, one value solution of $(S_2[0], S_2'[0])$ on average can be found by looking up `DDT`, and similarly for $(S_2[2], S_2'[2])$,
3. With $S_2[0,2]$ and $S_1[0,1]$, all byte values of S_1 and S_2, and hence S_3 are determined, again by Property 1.

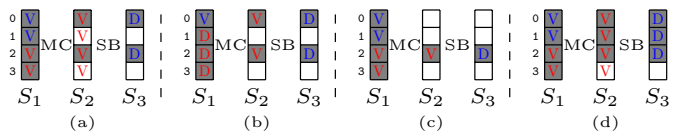


Fig. 21: Connecting differences with `DoF` from internal states

Property 3. For the case in Figure 21 (b) with known values $S_0[0]$ and $S'_0[0]$, known difference $\Delta S_3[0,2]$, and zero difference in $\Delta S_3[1,3]$, 4 equations with 6-bytes degree of freedom can be established, hence 2^{16} solutions are expected on average. We randomly guess the byte value $S_2[0]$, then $S'_2[0]$ can be computed as $\mathtt{SB}^{-1}(\mathtt{SB}(S_2[0]) \oplus \Delta S_3[0])$. Together with 2 zero-difference bytes $\Delta S_2[1,3]$ and known $\Delta S_1[0]$, all differences ΔS_1 and ΔS_2 can be deducted, and so the value $S_2[2]$ by looking up DDT with $(\Delta S_2[2], \Delta S_3[2])$. The overall time complexity is 1.

Property 4. For the case in Figure 21 (c) with known values $S_0[0,1]$ and $S'_0[0,1]$, known difference $\Delta S_3[2]$, and zero difference in $\Delta S_3[0,1,3]$, 4 equations with 4-byte degrees of freedom can be established, and one solution is expected on average. Since there are 5 input/output bytes of MC with known differences, i.e., $\Delta S_1[0,1]$ and $\Delta S_2[0,1,3]$, the system is over-defined and has solution with probability 2^{-8} due to Property 1. Then, both byte difference and value of $S_2[2]$ are determined. For each value of $S_2[0]$, all the values of S_2 are determined, and there are 2^8 possibilities of $S_2[0]$ leading to 2^8 solution pairs. Hence, for a random given S_1[V] and S_3[D], it is expected to find $2^{-8} \times 2^8 = 1$ solution on average. The average time complexity is 1.

Property 5. For the case in Figure 21 (d) with known values $S_0[0,1]$ and $S'_0[0,1]$, known difference $\Delta S_3[0,1,2]$, and zero difference in $\Delta S_3[3]$, 4 equations with 4-byte degrees of freedom can be established, and one solution is expected on average. We first guess one byte $S_2[0]$, the $S'_2[0]$ is deduced as $\mathtt{SB}^{-1}(\mathtt{SB}(S_2[0]) \oplus \Delta S_3[0])$, and all difference bytes of ΔS_2 including $\Delta S_2[1,2]$ are determined due to knowledge of $\Delta S_1[0,1]$ and $\Delta S_2[0,3]$ together with Property 1. By looking up DDT with $(\Delta S_2[1,2], \Delta S_3[1,2])$, values $S_2[1,2]$ are deduced. However, for MC between S_1 and S_2, 5 input/output bytes are known now, i.e., $S_1[0,1]$ and $S_2[0,1,2]$, the system is over-defined and consistent with probability 2^{-8}. Hence, $2^8 \times 2^{-8} = 1$ solution is expected with time complexity 1 per solution on average.

It is important to note that, other cases can be solved following a similar way from one of the 4 cases above. An application to Grøstl-512 hash function solves a 3-round inbound phase without qRAM.

We describe the steps solving the 3-round inbound phase in a more detailed way in Figure 22, 23 and 24. The first SH and last SH and MB are omitted. Let $S^{(i)}[j]$ be the jth byte of the ith column for state S. For given ΔS_1 and ΔS_7, as shown in Figure 22, in Step 1, we first guess the 32-byte (2^{256}) values S_6[V]. With ΔS_7, deduce S'_6[V]. We introduce Algorithm 9 to find the conforming pair.

Complexity. The time complexity of Algorithm 9 is about $2^{8\times32-3\times8+9\times8} = 2^{304}$. Given $(\Delta S_1, \Delta S_7)$, it is expected to find $2^{256-24+72-8\times4-16\times2-240} = 1$ conforming pair with 2^{16} memory cost to store DDT. At FSE 2012, Jean et al's [45] solved the 3-round inbound with time complexity 2^{280} and memory complexity 2^{64}. Our algorithm is very friendly to quantum implementation without qRAM together with Property 2, since Grøstl adopts the same Sbox with AES.

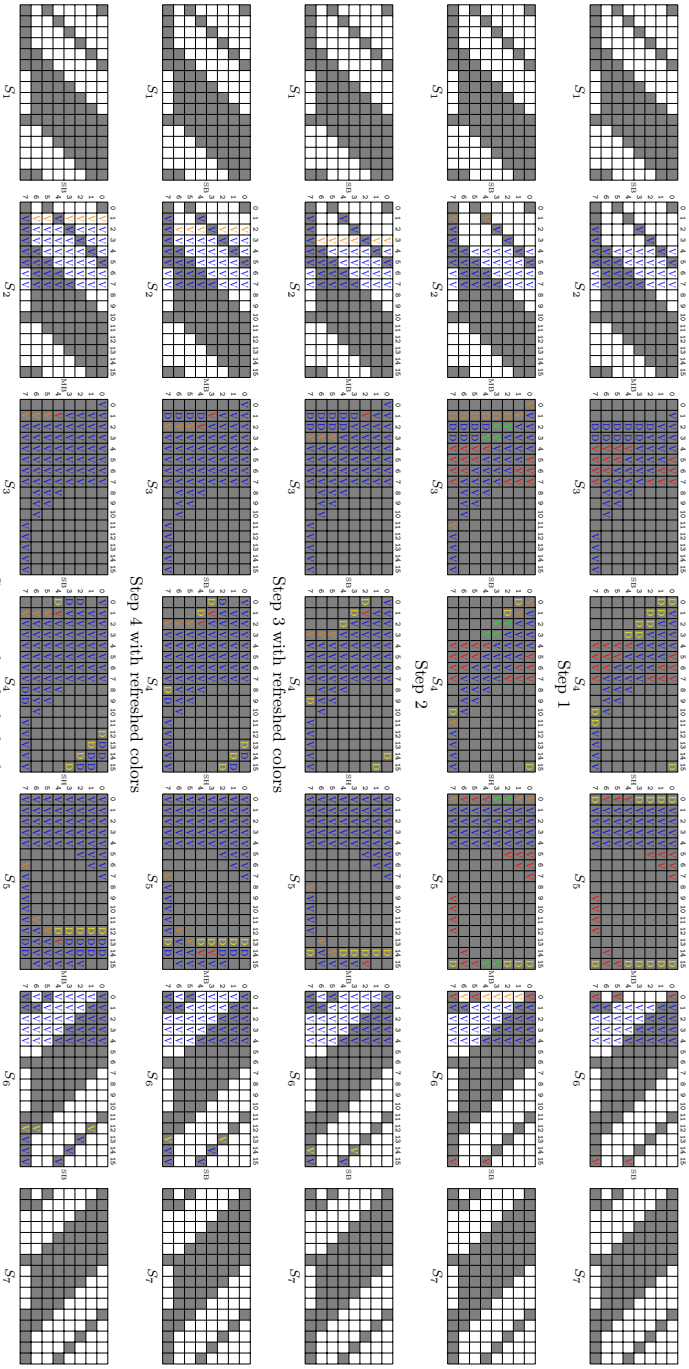


Fig. 22: Inbound Phase of Grøstl-512: Steps 1-5

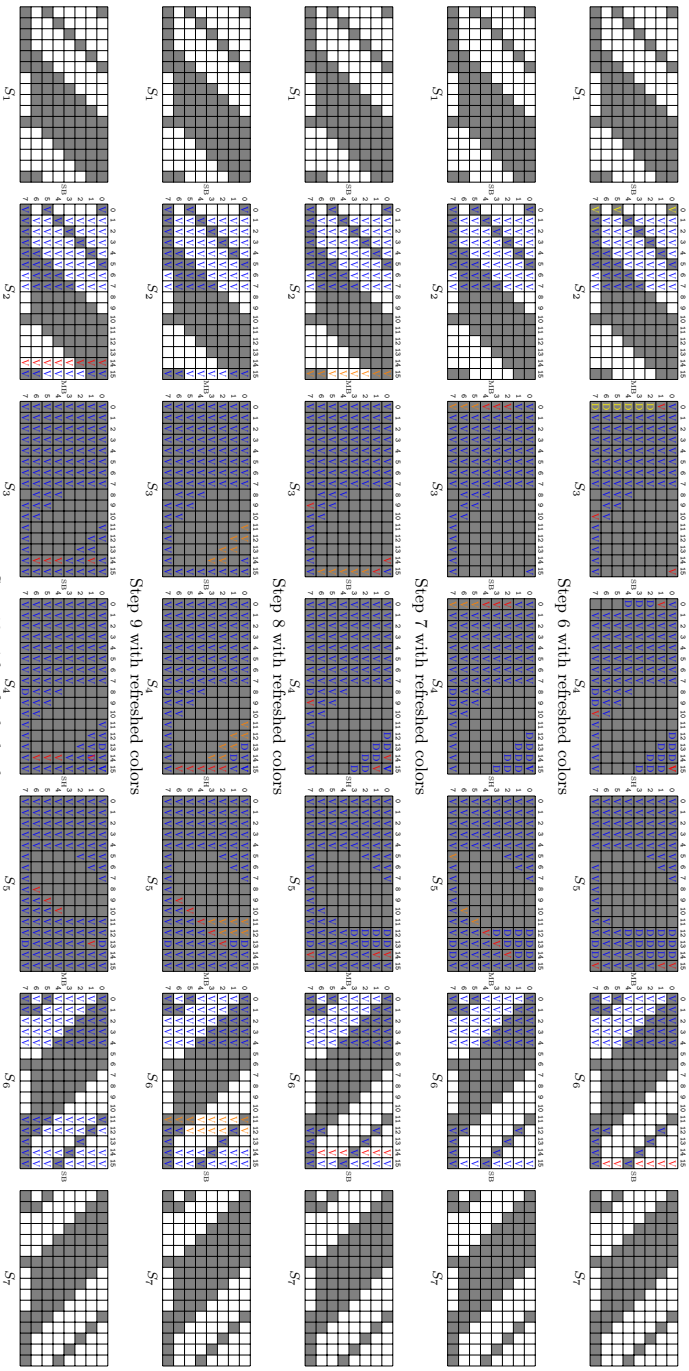


Fig. 23: Inbound Phase of Grøstl-512: Steps 6-10

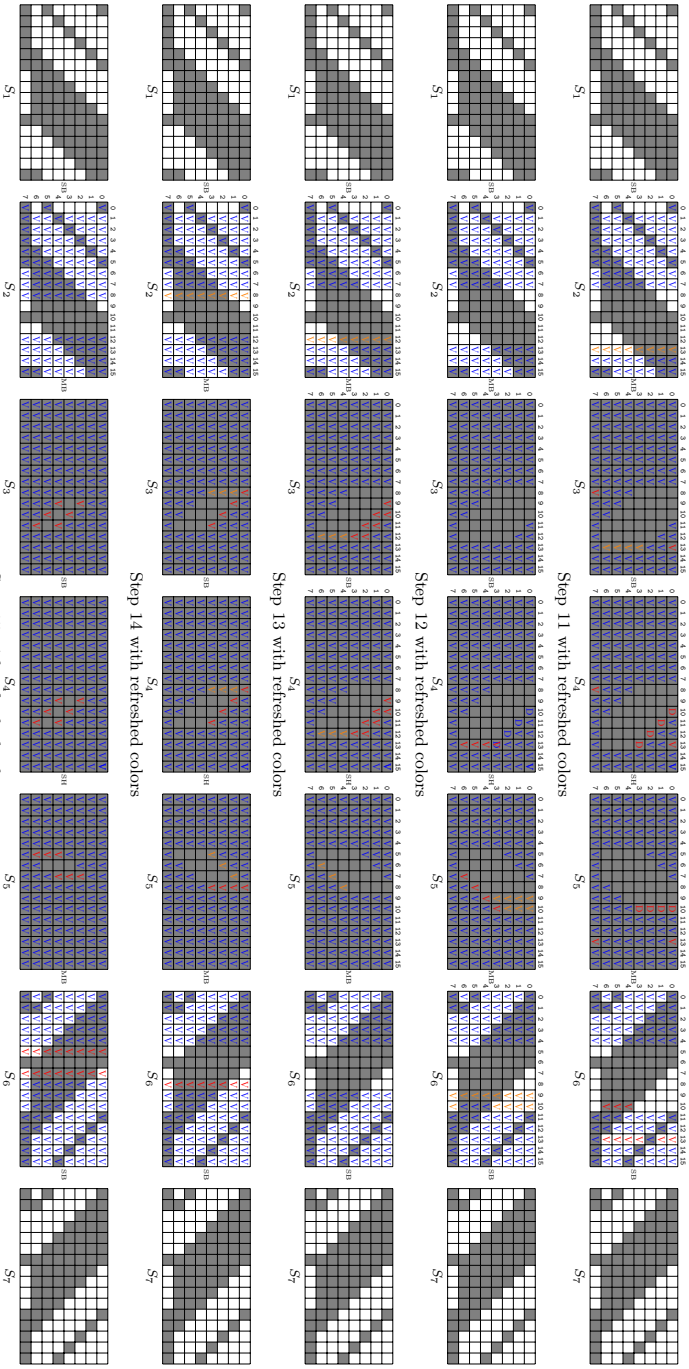


Fig. 24: Inbound Phase of Grøstl-512: Steps 11-15

Algorithm 9: Deduing the conforming pair with given $(\Delta S_1, \Delta S_7)$

for $S_6[V] \in \mathbb{F}_2^{8\times 32}$ *in Figure 22 (Step 1)* **do**
  Compute backward to S_3.
  if $(\Delta S_2^{(3)}, \Delta S_3^{(3)})$, $(\Delta S_2^{(4)}, \Delta S_3^{(4)})$, $(\Delta S_2^{(6)}, \Delta S_3^{(6)})$ *satisfy Pro. 1* **then**
    In Figure 22 (Step 1): Deduce $S_3^{(2)}[D]$ with 2-byte $S_3^{(2)}[V]$ and 6 nonactive bytes $S_2^{(2)}$ by Property 1. Together with ΔS_1, deduce $S_2^{(2)}[V]$; Similarly, deduce $S_2^{(3,5,7)}[V]$, $S_3^{(3)}[D]$ and $S_3^{(5,7)}[V]$
    for $(S_2^{(4)}[0], S_2^{(6)}[0], S_2^{(3)}[0], S_2^{(2)}[0], S_2^{(1)}[0], S_6^{(15)}[0], S_2^{(0)}[0], S_6^{(14)}[0], S_2^{(15)}[0]) \in \mathbb{F}_2^{8\times 9}$ **do**
      In Figure 22 (Step 1): Deduce $S_3^{(4,6)}[V]$ with guessed $(S_2^{(4)}[0], S_2^{(6)}[0])$; Deduce $S_5^{(15)}[D]$, $S_6^{(15)}[D]$ with 2-byte $S_5^{(15)}[V]$ and 6 inactive bytes in $S_6^{(15)}$; Deduce $S_5^{(0)}[D]$ and $S_5^{(0)}[V]$
      In Figure 22 (Step 2): Deduce $S_3[V]$ and $S_4[V]$ by DDT; Deduce $S_5^{(0)}[V]$, $S_6^{(0)}[V]$ by Properyty 1; Deduce $S_3^{(1)}[D]$, $S_2^{(1)}[V]$ by $S_3^{(1)}[0,1]$ and 6 inactivey bytes in $S_2^{(1)}$
      In Figure 22 (Step 3): Deduce $S_3[V]$, $S_4[V]$ by DDT; Deduce $S_2[V]$ and $S_3[V]$ with guessed $S_2^{(3)}[0]$; Deduce $S_5[D]$ and $S_6[V]$
      In Figure 22 (Step 4): Deduce $S_3[V]$ and $S_4[V]$; Deduce $S_2[V]$, $S_3[V]$ with guessed $S_2^{(2)}[0]$; Deduce $S_5[D]$, $S_6[V]$
      In Figure 22 (Step 5): Deduce $S_3[V]$; Deduce $S_2[V]$, $S_3[V]$ with guessed $S_2^{(1)}[0]$; Deduce $S_5[D]$, $S_6[V]$
      In Figure 22 (Step 6): Deduce $S_5[V]$, $S_6[V]$ with guessed $S_6^{15}[0]$; Deduce $S_2[V]$, $S_3[D]$with guessed $S_2^{(0)}[0]$
      In Figure 23 (Step 7): Deduce $S_3[V]$, $S_4[V]$ with DDT; Deduce $S_3[V]$ with `MB`;
      In Figure 23 (Step 8): Deduce $S_5[V]$, $S_6[V]$ with gessed $S_6^{(14)}[0]$; Deduce all the differences in $\Delta S_2^{(15)}$ and $\Delta S_3^{(15)}$ with known bytes $S_3^{(15)}[0,1,7]$ and the guessed $S_2^{(15)}[0]$; Deduce $S_3^{(15)}[2,3]$ with DDT; Deduce all values of this column with 5 known bytes $S_3^{(15)}[0,1,2,3,7]$ and 4 known bytes $S_2^{(15)}[0,1,6,7]$ (a filter of 2^{-8})
      In Figure 23 (Step 9): The red bytes in S_4 are deduced in Step 8; With the red and blue bytes, deduce $S_5[V]$, $S_6[V]$
      In Figure 23 (Step 10): with 4-byte $S_3^{(14)}[V]$ and 5 inactive bytes $S_3^{(14)}$, we first deduce all the differences of the same column, which is a filter of 2^{-8}; Then, with deduced difference $\Delta S_3^{(14)}[1]$ and known $\Delta S_4^{(14)}[1]$, get the value $S_3^{(14)}[1]$ by DDT; With values $S_3^{(14)}[0,1,2,3,7]$ and $S_2^{(14)}[0,1,2]$, deduce bytes marked by red
      In Figure 24 (Step 11): Deduce $S_5[V]$, $S_6[V]$; Deduce $S_5^{(10)}[D]$ with 4 $S_5^{(10)}[V]$ and 5 inactive bytes in $S_6^{(10)}$, which is also a filter of 2^{-8}; Deduce $S_2[V]$ and $S_3[V]$
      In Figure 24 (Step 12): Deduce $S_4[V]$ from S_3, where the known $\Delta S_4^{(13)}[3]$ acts as a filter of 2^{-8}; Deduce $S_5[V]$ and $S_6[V]$
      In Figure 24 (Step 13): We deduce $S_4[V]$ from S_5 of Step 12; Deduce $S_2[V]$, $S_3[V]$ from the blue and red bytes (a filter of 2^{-16})
      In Figure 24 (Step 14): Deduce $S_5[V]$, $S_6[V]$ with Property 5; Deduce $S_2[V]$, $S_3[V]$ (a filter of 2^{-16})
      In Figure 24 (Step 15): Deduce $S_5[V]$, $S_6[V]$; Deduce the last three bytes $S_3^{(9)}[3]$, $S_3^{(10)}[4]$, $S_3^{(11)}[5]$. Building similar equations to Equation (6), we deduce the filter in this step is $2^{-16-16-48-48-48-64} = 2^{-240}$

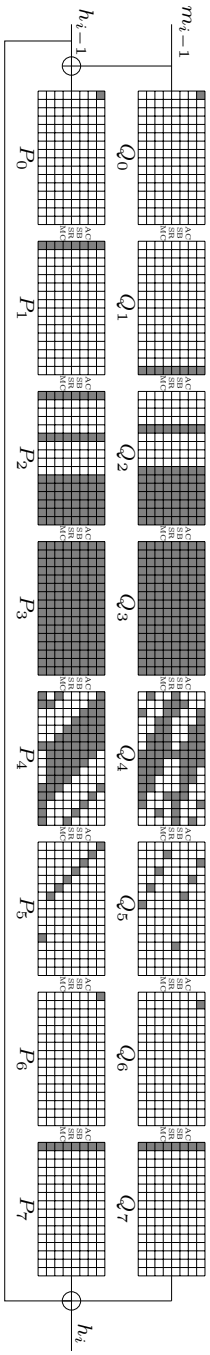


Fig. 25: Quantum Semi-free Start Collision on 7-round Grøstl-512

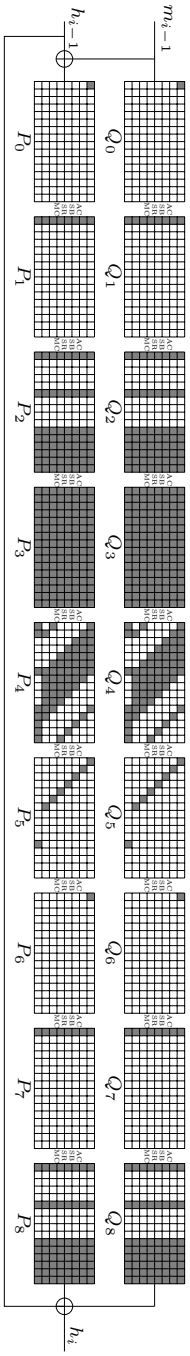


Fig. 26: Quantum Semi-free Start Collision on 8-round Grøstl-512 v0

G.1 Finding the Conforming Pair in Quantum Way

Given $(\Delta S_1, \Delta S_7)$, we convert Algorithm 9 into a quantum one. We use the quantum amplitude amplification algorithm. For Line 1 to Line 3 in Figure 22 (Step 1), we define $g : \mathbb{F}_2^{256} \mapsto \mathbb{F}_2$ as $g(\Delta S_1, \Delta S_7; x)$, where $x = S_6[\mathrm{V}] \in \mathbb{F}_2^{256}$. $g(\Delta S_1, \Delta S_7; x) = 1$ if and only if $(\Delta S_1, \Delta S_7; x)$ satisfy the condition in Line 3, i.e., check if the computed $(\Delta S_2^{(3)}, \Delta S_3^{(3)})$, $(\Delta S_2^{(4)}, \Delta S_3^{(4)})$ and $(\Delta S_2^{(6)}, \Delta S_3^{(6)})$ satisfy Property 1, where the probability is 2^{-24}. Let $T_{(\Delta S_1, \Delta S_7)} := \{x : x \in \mathbb{F}_2^{256}, g(\Delta S_1, \Delta S_2; x) = 1\}$, whose size is $|T_{(\Delta S_1, \Delta S_7)}| = 2^{256-24} = 2^{232}$. Then, $\mathcal{A}$ is to apply Grover's algorithm to $g(\Delta S_1, \Delta S_7; \cdot)$ to produce the superposition state $|\phi\rangle = \sum_{x \in T_{(\Delta S_1, \Delta S_7)}} |x\rangle$. Hence, the time complexity $|\mathcal{A}|_{RT} \approx \sqrt{2^{24}} = 2^{12}$.

The projector $\mathcal{P}$ is to identify the good state $|x\rangle$, such that it leads to a good pair conforming to $(\Delta S_1, \Delta S_7)$. Given $|x\rangle$, we traverse $(S_2^{(4)}[0], S_2^{(6)}[0], S_2^{(3)}[0], S_2^{(2)}[0], S_2^{(1)}[0], S_6^{(15)}[0], S_2^{(0)}[0], S_6^{(14)}[0], S_2^{(15)}[0]) \in \mathbb{F}_2^{72}$ in Line 5 of Algorithm 9 to check if $|x\rangle$ leads to the good pair. We define $f : \mathbb{F}_2^{72} \mapsto \mathbb{F}_2$ as $f(\Delta S_1, \Delta S_7, x; y)$, where $y = (S_2^{(4)}[0], S_2^{(6)}[0], S_2^{(3)}[0], S_2^{(2)}[0], S_2^{(1)}[0], S_6^{(15)}[0], S_2^{(0)}[0], S_6^{(14)}[0], S_2^{(15)}[0]) \in \mathbb{F}_2^{72}$. $f(\Delta S_1, \Delta S_7, x; y) = 1$ if and only if $(\Delta S_1, \Delta S_7, x; y)$ leads to a good pair. The quantum oracle $\mathcal{U}_{\mathcal{P}}$ of the projector $\mathcal{P}$ is to apply Grover's algorithm to $f(\Delta S_1, \Delta S_7, x; \cdot)$ to find the good one. The complexity $|\mathcal{U}_{\mathcal{P}}|_{RT} \approx \sqrt{2^{72}} = 2^{36}$.

In the superpostion $|\phi\rangle = \sum_{x \in T_{(\Delta S_1, \Delta S_2)}} |x\rangle$ derived by applying $\mathcal{A}$, the amplitude α of the good $|x\rangle$ that leads to a good pair conforming $(\Delta S_1, \Delta S_2)$ is 2^{-232}. Hence, according to Equation (2), the totall complexity of is about $\sqrt{2^{232}} \cdot (2^{12} + 2^{36}) \approx 2^{152}$.

If we convert Jean et al.'s method [45] into a quantum one, a 2^{64} qRAM will be needed with time complexity of 2^{140}. In the Time $\times$ Space scenario, our method reduces the overal complexity by $2^{140+64-152} = 2^{52}$, which essentially makes our quantum semi-free-start collision on 7-round `Grøstl-512` possible.

G.2 7-round Quantum Semi-free Start Collision Attack

As shown in Figure 25, we perform the rebound attacks in both P and Q permutations. In P permutation, the 3-round inbound phase covers P_2 to P_5. The outbound phase includes P_2 to P_0 and P_5 to P_7, whose probability is 2^{-112}. Given $(\Delta P_2, \Delta P_5) \in \mathbb{F}_2^{64+64}$, let $F : \mathbb{F}_2^{128} \mapsto \mathbb{F}_2$. $F_P(\Delta P_2, \Delta P_5) = 1$ if and only if the pair generated by solving the 3-round inbound part with given $(\Delta P_2, \Delta P_5)$ satisfies the outbound part. Applying Grover's algorithm to $F_P(\cdot)$, we get the superposition $|\rho\rangle = \sum_{F_P(\Delta P_2, \Delta P_5)=1} |\Delta P_2, \Delta P_5\rangle$. The time complexity is about $\sqrt{2^{112}} \cdot 2^{152} = 2^{208}$. Similarly, we define function F_Q and get $|\sigma\rangle = \sum_{F_Q(\Delta Q_2, \Delta Q_5)=1} |\Delta Q_2, \Delta Q_5\rangle$ with time 2^{208}.

We perform the amplitude amplification algorithm. Let $\mathcal{A}$ send the $|0\rangle$ to $|\rho\rangle \otimes |\sigma\rangle$. With the above analysis, we know $|\mathcal{A}|_{RT} \approx 2^{208}$. The unitary operator $\mathcal{U}_{\mathcal{P}}$ check if $|\Delta P_2, \Delta P_5\rangle \otimes |\Delta Q_2, \Delta Q_5\rangle$ produce a semi-free start collision, whose probability is 2^{-16}. $|\mathcal{U}_{\mathcal{P}}|_{RT} \approx 1$. Hence, the total time is $\sqrt{16} \cdot (2^{208} + 1) \approx 2^{214}$.

G.3 Quantum Semi-free Start Collision Attack on 8-round Grøstl-512 v0

Similarly, we build quantum semi-free start collision attack on 8-round Grøstl-512 v0. As shown in Figure 26, the probability of the outbound phase is also 2^{-112}. Hence, $|\mathcal{A}|_{RT}$ is also 2^{208}. However, after generating $|\rho\rangle \otimes |\sigma\rangle$, the probability of $\Delta Q_0 = \Delta P_0$ and $\Delta Q_8 = \Delta P_8$ is 2^{-72}. Hence, the total complexity is $\sqrt{2^{72}} \cdot (2^{208} + 1) = 2^{244}$.