Quantum Attacks on Hash Constructions with Low Quantum Random Access Memory

Xiaoyang Dong[1,2,6,7(✉)], Shun Li[3(✉)], Phuong Pham[3(✉)], and Guoyan Zhang[4,5,7(✉)]

[1] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China
xiaoyangdong@tsinghua.edu.cn
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China
[3] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore
shun.li@ntu.edu.sg, pham0079@e.ntu.edu.sg
[4] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China
guoyanzhang@sdu.edu.cn
[5] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
[6] Zhongguancun Laboratory, Beijing, China
[7] Shandong Institute of Blockchain, Jinan, China

Abstract. At ASIACRYPT 2022, Benedikt, Fischlin, and Huppert proposed the quantum herding attacks on iterative hash functions for the first time. Their attack needs exponential quantum random access memory (qRAM), more precisely $2^{0.43n}$ quantum accessible classical memory (QRACM). As the existence of large qRAM is questionable, Benedikt et al. leave an open question on building low-qRAM quantum herding attacks.

In this paper, we answer this open question by building a quantum herding attack, where the time complexity is slightly increased from Benedikt et al.'s $2^{0.43n}$ to ours $2^{0.46n}$, but it does not need qRAM anymore (abbreviated as no-qRAM). Besides, we also introduce various low-qRAM or no-qRAM quantum attacks on hash concatenation combiner, hash XOR combiner, Hash-Twice, and Zipper hash functions.

Keywords: Quantum computation · qRAM · Herding Attack · Hash Combiner

1 Introduction

Shor's seminal work [59] shows that sufficiently large quantum computers allow factorization of large numbers and computation of discrete logarithms in polynomial time, potentially dooming many public-key schemes in use today. To

J. Guo and R. Steinfeld (Eds.): ASIACRYPT 2023, LNCS 14440, pp. 3–33, 2023.
https://doi.org/10.1007/978-981-99-8727-6_1

meet the challenges posed by quantum computers, the public-key cryptography community and standardization organizations have invested a lot of effort in the research of post-quantum public-key schemes. In particular, NIST has initiated a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptography algorithms [55]. For symmetric cryptography, the community has also recently witnessed many important quantum cryptanalysis results [15–17,25,38,41,48] since the initial work of Kuwakado and Morii, who showed that the classically provably secure Even-Mansour cipher and the three-round Feistel network can be broken in polynomial time with the help of quantum computers [46,47]. Most of these attacks that enjoy exponential speedup rely on Simon's algorithm [60] to find a key-dependent hidden period where access to a quantum superposition oracle of key primitives is necessary. This is a fairly strong (computation) model, and its actual relevance is sometimes questioned [13]. Therefore, a more complex attack still makes sense if it does not require online queries to the superposition oracles of the keyed primitives [13,14,18,35,57].

For keyless primitives, especially hash functions, quantum attacks are easier to launch, since there is no need for online queries and all computations can be done offline. The classical algorithm finds collisions of n-bit output hash functions with time complexity $\mathcal{O}(2^{n/2})$. In the quantum setting, the BHT algorithm [20] finds collisions with a query complexity of $\mathcal{O}(2^{n/3})$ if $\mathcal{O}(2^{n/3})$ quantum random access memory (qRAM) is available. However, it is generally acknowledged that the difficulty of fabricating large qRAMs is enormous [31,32]. So quantum algorithms (even has relatively high time complexity) using less or no qRAM is desirable. At ASIACRYPT 2017, Chailloux, Naya-Plasencia and Schrottenloher first overcome the $\mathcal{O}(2^{n/2})$ classical bound without using qRAM [21]. The time complexity of the algorithm is $\mathcal{O}(2^{2n/5})$, and the classical memory is $\mathcal{O}(2^{n/5})$. Also, a quantum algorithm for the generalized birthday problem (or the k-XOR problem) in settings with and without large qRAM can be found in [33,53]. Besides the generic attacks on hash functions, the first dedicated quantum attack on hash functions was presented at EUROCRYPT 2020 by Hosoyamada and Sasaki [36], showing quantum attacks on AES-MMO and Whirlpool by exploring differentials whose probability is too low to be useful in the classical setting. Later, refined collision and preimage attacks on hash functions have been presented subsequently by Dong et al. [26–28], Flórez Gutiérrez et al. [29], Hosoyamada and Sasaki [37], Schrottenloher and Stevens [58].

The Merkle-Damgård construction [22,52] is a popular way to build hash functions, where a single compression function is iteratively called to extend the input domain from a fixed length to arbitrary length and the digest length is usually the same as that of the internal state. However, some widely deployed hash function standards (such as MD5 and SHA-1) based Merkle-Damgård construction have been broken [61–63]. Besides, Kelsey and Schneier [43] have demonstrated a generic second-preimage attack against all hash functions based on the classical Merkle-Damgård construction, when the challenge message is long. At CRYPTO 2004, Joux [40] introduced multi-collision attacks on iterated hash

functions. At EUROCRYPT 2006, Kelsey and Kohno introduced the herding attack (also known as nostradamus attack) [42], in which the adversary commits to a hash value T of an iterated hash function $\mathcal{H}$, such that when later given a message prefix P, the adversary is able to find a suitable "suffix explanation" S with $\mathcal{H}(P\|S) = T$.

In order to obtain a more secure hash function, and to ensure compatibility, researchers and developers try to combine the output of two (or more) independent hash functions to provide better security in case one or even both hash functions are weak. Practical examples can be found in TLS [23] and SSL [30]. There are several common hash combiners, such as concatenation combiner [56], XOR combiner, Hash-Twice [3], and Zipper hash [50]. However, the security of these hash combiners has also been challenged. At CRYPTO 2004, Joux [40] revealed that the concatenation combiner provides at most $n/2$-bit security for collision resistance and n-bit security for preimage resistance. Leurent and Wang [49] and Dinur [24] showed that the combiners may be weaker than each hash function. Besides, various cryptanalysis results [2–4,6,8,51] have been achieved on the hash combiners.

At ASIACRYPT 2022, Benedikt, Fischlin, and Huppert [9] considered quantum nostradamus attacks on iterative hash functions for the first time, and realized attacks of complexity $\mathcal{O}(2^{3n/7})$. The attack requires exponentially large qRAM, which is inherited from the BHT algorithm [20]. Since fabricating large qRAMs is difficult to realize [31,32], Benedikt et al. [9] left open questions for building low-qRAM quantum herding attack. In 2022, Bao et al. [7] built a low-qRAM quantum herding attack based Chailloux et al.'s multi-target preimage algorithm [21]. However, we find their algorithm is flawed and incorrect when building diamond structure for herding[1]. Therefore, the question is still open. Besides the quantum herding attack, Bao et al. also gave some quantum attacks on hash XOR and concatenation combiners, including collision, preimage, and herding attacks [7].

Our Contributions

In this paper, **for the first contribution,** we answer the open question by Benedikt et al. [9] to build the first valid low-qRAM quantum herding attack on iterated hash functions. We first convert Benedikt et al.'s quantum diamond-building algorithm (it needs exponential qRAM, i.e., $2^{3n/7}$ quantum accessible classical memory (QRACM)) into an algorithm that does not need qRAM anymore. The new algorithm is highly based on Chailloux et al.'s collision finding algorithm [21] with various adaptions. In our herding attack, we choose the leaves of the diamond structure to be prefixed with r-bit zeros, then apply Chailloux et al.'s collision finding to find the linking message S such that $H(P\|S)$ hits one of the leaves of the diamond structure. Note a previous work by Bao et al. [7] also built a quantum herding attack. However, in their attack, the Chailloux et al.'s

[1] Please find the detailed comments on Bao et al's attacks in Appendix A and B.

multi-target preimage algorithm [21] is applied, which can not take the advantage of the ability that attacker can choose the prefixed leaves of the diamond structure.

As the Second Contribution, for the quantum preimage attack on hash XOR combiners, we introduce an efficient low-qRAM quantum algorithm to build Leurent and Wang's interchange structure [49]. Then, based on Schrottenloher and Stevens's quantum Meet-in-the-Middle attack [58], or Ambainis' element distinctness algorithm [1], or Jaques and Schrottenloher's golden collision finding algorithm [39], we propose three different low-qRAM quantum preimage attacks on hash XOR combiner. Especially, our attack based on Jaques and Schrottenloher's method [39] reduces the $2^{0.143n}$ qubits of previous attack [7] to ours $2^{0.013n}$ qubits, without quantum accessible quantum memory (QRAQM). Moreover, the time complexity is also reduced from previous $2^{0.495n}$ to ours $2^{0.493n}$.

For hash concatenation combiner, we introduce a no-qRAM quantum collision attack and a no-qRAM quantum herding attack. In [7], both attacks need $2^{0.143n}$ qubits or $2^{0.333n}$ QRAQM . However, our attacks do not need qRAM and the number of qubits needed is also of polynomial size. We also introduce quantum herding attacks on other important hash combiners, including Hash-Twice, and Zipper hash function, by exploiting their different features. All the attacks are summarized in Table 1.

Table 1. A Summary of the Attacks. QRACM: quantum accessible classical memory, QRAQM: quantum accessible quantum memory, cRAM: classical random access memory

Target	Attacks	Settings	Time	Qubits	QRACM	QRAQM	cRAM	Generic	Ref.
$\mathcal{H}$	Herding	Classical	$2^{0.67n}$	-	–	–	$2^{0.67n}$	–	[42]
		Quantum	$2^{0.43n}$	$\mathcal{O}(n)$	$2^{0.43n}$	–	–	–	[9]
		Quantum	$2^{0.46n}$	$\mathcal{O}(n)$	–	–	$2^{0.23n}$	–	Sect. 4
$\mathcal{H}_1 \oplus \mathcal{H}_2$	Preimage	Classical	$2^{0.83n}$	-	-	-	$2^{0.33n}$	2^n	[49]
		Classical	$2^{0.67n}$	–	–	-	–	2^n	[24]
		Classical	$2^{0.612n}$	–	–	–	$2^{0.61n}$	2^n	[6]
		Quantum	$2^{0.476n}$	$\mathcal{O}(n)$	-	$2^{0.333n}$	-	$2^{0.5n}$	[7]
		Quantum	$2^{0.495n}$	$2^{0.143n}$	$2^{0.033n}$	–	$2^{0.2n}$	$2^{0.5n}$	[7]
		Quantum	$2^{0.493n}$	$2^{0.013n}$	$2^{0.047n}$	–	$2^{0.2n}$	$2^{0.5n}$	Sect. 5.3
		Quantum	$2^{0.485n}$	$\mathcal{O}(n)$	$2^{0.057n}$	$2^{0.0285n}$	$2^{0.2n}$	$2^{0.5n}$	Sect. 5.3
		Quantum	$2^{0.485n}$	$\mathcal{O}(n)$	$2^{0.043n}$	$2^{0.0285n}$	$2^{0.2n}$	$2^{0.5n}$	Sect. 5.3
$\mathcal{H}_1 \| \mathcal{H}_2$	Collision	Classical	$2^{0.5n}$	–	–	–	–	2^n	[40]
		Quantum	$2^{0.333n}$	$\mathcal{O}(n)$	–	$2^{0.333n}$	–	$2^{0.67n}$	[7]
		Quantum	$2^{0.43n}$	$2^{0.143n}$	–	–	$2^{0.2n}$	$2^{0.67n}$	[7]
		Quantum	$2^{0.4n}$	$\mathcal{O}(n)$	–	–	$2^{0.2n}$	$2^{0.67n}$	Sect. 6
	Herding	Classical	$2^{0.67n}$	–	–	–	$2^{0.33n}$	–	[3]
		Quantum	$2^{0.444n}$	$\mathcal{O}(n)$	–	$2^{0.333n}$	–	–	[7]
		Quantum	$2^{0.49n}$	$2^{0.143n}$	-	-	$2^{0.2n}$	-	[7]
		Quantum	$2^{0.467n}$	$\mathcal{O}(n)$	–	–	$2^{0.2n}$	–	Sect. 7
Hash-Twice	Herding	Classical	$2^{0.667n}$	-	-	-	$2^{0.33n}$	–	[3]
		Quantum	$2^{0.467n}$	$\mathcal{O}(n)$	–	–	$2^{0.2n}$	-	Sect. 8
Zipper	Herding	Classical	$2^{0.667n}$	–	–	–	$2^{0.33n}$	–	[3]
		Quantum	$2^{0.467n}$	$\mathcal{O}(n)$	–	–	$2^{0.2n}$	-	Sect. 9

2 Preliminaries

2.1 Quantum Computation and Quantum RAM

The state of the n-qubit quantum system can be described as the unit vector $\{|i\rangle : 0 \leq i < 2^n\}$ in $\mathbb{C}^{2^n}$ under the orthogonal basis. Quantum algorithms are typically implemented by manipulating the state of an n-qubit system through a series of unitary transformations and measurements, where all unitary transformations can be implemented as a series quantum gates in *quantum circuit models* [54]. The efficiency of a quantum algorithm is quantified based on the number of quantum gates used.

Superposition Oracles for Classical Circuit. Let the quantum oracle of a function $f : \mathbb{F}_2^m \mapsto \mathbb{F}_2^n$ be 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 $y \in \mathbb{F}_2^n$. When $\mathcal{U}_f$ acts on superposition states, we have

$$\mathcal{U}_f \left(\sum_{x \in \mathbb{F}_2^n} a_i |x\rangle |y\rangle \right) = \sum_{x \in \mathbb{F}_2^n} a_i |x\rangle |y \oplus f(x)\rangle . \tag{1}$$

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

- *Setup* operation, i.e., sampling a uniform element from X. Denote the cost (execution time) of *Setup* as $|Setup|_{RT}$.
- *Checking* operation, i.e. checking if an element is labeled. Denote the cost (execution time) of *Checking* as $|Checking|_{RT}$.

Grover's algorithm [34] is a quantum search process for finding the labeled elements, whose complexity is a function of the quantum *Setup* cost $|Setup|_{RT}$ of construction of uniform superposition of all elements from X, and the quantum *Checking* cost $|Checking|_{RT}$. The time complexity of Grover's algorithm is $\sqrt{1/\epsilon} \cdot (|Setup|_{RT} + |Checking|_{RT})$. Assuming the *Setup* and *Checking* steps are simple, Grover's algorithm can find the element $x \in M$ at a cost of $\mathcal{O}(\sqrt{1/\epsilon})$.

Grover's algorithm can also be described as a special case of quantum amplitude amplification (QAA), which is a quantum algorithm introduced by Brassard, Høyer, Mosca, and Tapp [19]. Intuitively, assuming there exists an quantum algorithm $\mathcal{A}$ to produce a superposition of the good subspace and the bad subspace of X. Let a be the initial success probability that the measurement of $\mathcal{A}|0\rangle$ is good. Let $\mathcal{B}$ be a function that classifies the outcomes of $\mathcal{A}$ as either good or bad state. Quantum Amplitude Amplification (QAA) technique achieves the same result as Grover's algorithm with a quadratic improvement. The time complexity of QAA is about

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

Quantum Random Access Memories (qRAM) can be conceptualized as the quantum counterpart of classical random access memory (RAM). In the classic setup, RAM facilitates access (read and write operations) to memory elements in time $\mathcal{O}(1)$ regardless of storage size. Following [45,58], qRAM comes in two flavors: Quantum Accessible Classical Memory (QRACM), which enables access to classical data in quantum superposition; and Quantum Accessible Quantum Memory (QRAQM), where data is stored in quantum memory. Consider a scenario where we intend to store a list of data, denoted as $D = (x_0, x_1, \cdots, x_{2^k-1})$, with each x_i representing an n-bit data. In this context, the qRAM for accessing the data D is established as a quantum gate. This qRAM is defined through a unitary operator $U_{qRAM}(D)$, which is expressed as follows:

$$U_{qRAM}(D) : |i\rangle |x_0, x_1, \cdots, x_{2^k-1}\rangle |y\rangle \rightarrow |i\rangle |x_0, x_1, \cdots, x_{2^k-1}\rangle |y \oplus x_i\rangle ,$$

Here, i takes values from the set $\{0,1\}^k$, and y represents an n-bit value. In both QRACM and QRAQM, we assume that this gate costs $\mathcal{O}(1)$. For QRACM, i is superposed but the x_i are classical; For QRAQM, both i and x_i are superposed. For example, the BHT collision finding algorithm [20] requires QRACM, the quantum element distinctness [1] and quantum meet-in-the-middle attack [58] require QRAQM. Obviously, QRAQM is the strongest quantum memory model.

For the time being, it is unknown how a working qRAM (at least for large qRAMs) can be built. Nevertheless, this disappointing fact does not stop researchers from working in a model where large qRAMs are available, in the same spirit that people started to work on classical and quantum algorithms long before a classical or quantum computer had been built. From another perspective, the absence of large qRAMs makes it quite meaningful to conduct research in an attempt to reduce or even avoid the use of qRAM in quantum algorithms.

Quantum Element Distinctness Problem

Problem 1. Given a set $S = \{x_1, x_2, \cdots, x_N\}$, does it exist i, j such that $1 \leq i < j \leq N$ and $x_i = x_j$? If yes, return i, j.

In 2007, Ambainis proposed the quantum walk algorithm for the element distinctness problem [1] and achieved time complexity of $\mathcal{O}(N^{2/3})$ with $\mathcal{O}(N^{2/3})$ QRAQM. At SAC 2020, Jaques and Schrottenloher [39] solved the element distinctness problem (or golden collision problem by [39]) in the plain quantum circuit model (i.e., the computation is a sequence of basic quantum gates applied to a pool of qubits) in time complexity of $\mathcal{O}(N^{6/7})$ with $\mathcal{O}(N^{2/7})$ qubits without qRAM.

CNS Collision Finding Algorithm [21]**.** At ASIACRYPT 2017, Chailloux, Naya-Plasencia and Schrottenloher [21] introduced the first quantum collision finding algorithm without any qRAM. Their algorithm is denoted as CNS algorithm in this paper. The time complexity of the algorithm is $\mathcal{O}(2^{2n/5})$, with a classical memory of $\mathcal{O}(2^{n/5})$. The CNS algorithm is based on a quantum membership algorithm.

Definition 1. *Given a set L of 2^k n-bit strings, a classical membership oracle is a function f_L that computes: $f_L(x) = 1$ if $x \in L$ and 0 otherwise.*

A quantum membership oracle for L is an operator O_L that computes f_L:

$$O_L(|x\rangle\,|b\rangle) = |x\rangle\,|b \oplus f_L(x)\rangle\,.$$

When the set L of size 2^k is stored in some classical memory, Chailloux et al. implement the quantum operator O_L with $n2^k$ simple operations and $2n + 1$ qubits. Since in the following, the time complexity is number of queries of the compression functions of hash function, the $n2^k$ simple operations are then recorded as $\mathcal{O}(2^k)$ time complexity. The CNS collision finding algorithm can be divided into two parts, i.e., the precomputing part and the matching part.

Precomputing Part: Given a hash function h that $h(m) = T$, the CNS algorithm first builds a table L of size 2^k, where the r-bit most significant bits (MSB) of all $x \in L$ are zero, and store L in a classical memory. The way to build L is to perform 2^k times of Grover's algorithm with time complexity of $2^k \times 2^{r/2} = 2^{k+r/2}$.

The Matching Part: Apply the QAA algorithm. In the setup phase $\mathcal{A}$, the Grover's algorithm is applied to produce a superposition of m, where the r-bit MSBs of m are zero. The time of the setup phase is $|\mathcal{A}|_{RT} = 2^{r/2}$. Then, in the checking phase $\mathcal{B}$, a quantum membership algorithm is applied to classify that if m is in L or not. $|\mathcal{B}|_{RT} = 2^k$. Since the initial probability, that the measurement of $\mathcal{A}\,|0\rangle$ is good, is $a = \frac{2^k}{2^{n-r}}$ (since only the last $n - r$ bits should be matched). According to Eq. (2), time complexity of this part is

$$\sqrt{\frac{2^{n-r}}{2^k}} \cdot (2^{r/2} + 2^k). \tag{3}$$

Totally, the time of the CNS algorithm is

$$\sqrt{\frac{2^{n-r}}{2^k}} \cdot (2^{r/2} + 2^k) + 2^{k+r/2}. \tag{4}$$

By assigning $r = 2k = 2n/5$, the complexity given in Eq. (4) will be optimal, which is $\mathcal{O}(2^{2n/5})$. The number of qubits used is $\mathcal{O}(n)$. The classical memory is $2^{n/5}$ to store L.

In this paper, the CNS algorithm is frequently used. In several applications of our paper, only the **Matching Part** of the CNS algorithm is used with a given L, while L may be built in a different way than the **Precomputing Part** and thus have a different complexity than $2^{k+r/2}$. For example, in our quantum herding attack in Sect. 4, the time to build L is the time to build the diamond structure. Therefore, the time complexity of the **Matching Part** should be weighed against the different time complexity of constructing L. To use the CNS algorithm more flexibly, we define the **Matching Part** as $\mathrm{CNS}_h(m, L)$ in Definition 2 for a given table L and h in the following.

Definition 2. *Let $CNS_h(m, L)$ be the matching part of CNS algorithm, which finds m so that $h(m) \in L$. Given the table L of size 2^k stored in classical memory, whose elements are prefixed with r-bit zeros, the time complexity* $|CNS_h(m, L)|_{RT} = \sqrt{\frac{2^{n-r}}{2^k}} \cdot (2^{r/2} + 2^k)$.

Quantum Meet-in-the-Middle Algorithm. At CRYPTO 2022, Schrottenloher and Stevens [58] applied the quantum two-list merging algorithm to build the quantum MitM attack: For a given global guess $G \in \mathbb{F}_2^g$, two small lists are computed and merged to on the fly. Suppose the two small lists are L_1 and L_2, the goal is to determine if there are elements $x \in L_1$ and $y \in L_2$ such that $x = y$ (called a solution). Let $O_{\texttt{merge}}$ be the unitary operator that

$$O_{\texttt{merge}}(|G\rangle\,|b\rangle) = |G\rangle\,|b \oplus f(G)\rangle\,, \text{where } f(G) = \begin{cases} 1 \text{ if a solution occurs} \\ 0 \text{ otherwise} \end{cases}. \quad (5)$$

Lemma 1. *[58] Assume that there exists an implementation of $O_{\texttt{merge}}$ with time complexity T. Then there is a quantum MitM attack with time complexity:*

$$(\frac{\pi}{4}2^{g/2} + 1) \times T. \quad (6)$$

The T is roughly estimated by

$$\min(|L_1|, |L_2|) + \sqrt{\max(|L_{\texttt{merge}}|, |L_1|, |L_2|)}, \quad (7)$$

where $L_{\texttt{merge}}$ is the merged list. The QRAQM needed is of size $\min(|L_1|, |L_2|)$.

2.2 Iterated Hash Constructions

Iterated hash functions $\mathcal{H}(IV, M) = T$ commonly first pad and split the message M into message blocks of fixed length, i.e., $M = m_1 \| m_2 \| \cdots \| m_L$. The message blocks are processed sequentially and iteratively by the compression function h, i.e., $x_i = h(x_{i-1}, m_i)$, where $x_0 = IV$ is a public value, $T = x_L$ is the n-bit digest, the chaining value $x_i \in \mathbb{F}_2^n$. Two commonly used iterated hash constructions are the Merkle-Damgård construction [22,52] and the HAIFA construction [11]. In this paper, we only consider the Merkle-Damgård construction and its extensions.

The concatenation combiner $\mathcal{H}_1(IV_1, M) \| \mathcal{H}_2(IV_2, M) = T_1 \| T_2$ is one of the most studied hash combiner, which is first described by Preneel in 1993 [56]. In 2004, Joux [40] described the multi-collision attack on the $2n$-bit output hash combiner with $2^{n/2}$ time complexity for collision attack and 2^n time complexity for preimage attack. Besides the concatenation combiner, there are other constructions:

- The XOR hash combiner $\mathcal{H}_1(IV_1, M) \oplus \mathcal{H}_2(IV_2, M) = T$.
- Hash-Twice is originally defined in [3]: $\mathcal{H}_2(\mathcal{H}_1(IV, M), M) = T$ shown in Fig. 1.
- Zipper hash [50] is defined as $\mathcal{H}_2(\mathcal{H}_1(IV, M), \overleftarrow{M}) = T$ shown in Fig. 2.

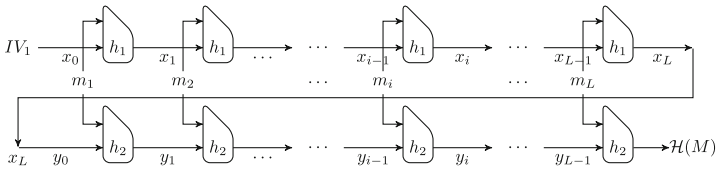


Fig. 1. Hash-Twice Construction

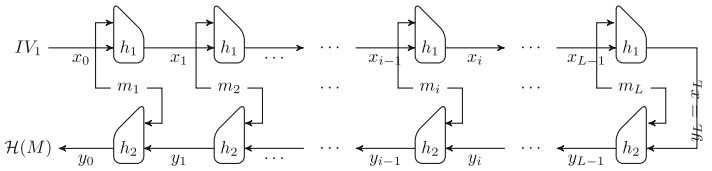


Fig. 2. Zipper Hash Construction

3 Basic Techniques and Their Quantum Versions

In this section, we give brief introductions of Joux's multi-collision technique, diamond structure (DS) and their quantum versions.

3.1 Joux's Multi-collision

At CRYPTO 2004, Joux [40] introduced an efficient method to build multi-collision on iterated hash functions. As shown in Fig. 3, started from x_0, the attacker performs t birthday attacks to find t collisions. Based on the message blocks $m_1, m_2, \cdots m_t$ and $m'_1, m'_2, \cdots m'_t$, the attacker can build 2^t collision message pairs (denoted as 2^t-$\mathcal{M}_{\mathtt{MC}}$), e.g., $(m_1 \| m'_2 \| \cdots \| m_t, m'_1 \| m_2 \| \cdots \| m'_t,)$. The time complexity to build the 2^t collision message pairs is $t \cdot 2^{n/2}$. In quantum setting, Bao et al. [7] first applied CNS's algorithm to build Joux's multi-collision, where one collision is built in time $2^{2n/5}$. Therefore, the time to build 2^t-$\mathcal{M}_{\mathtt{MC}}$ is $t \cdot 2^{2n/5}$. The quantum attack only uses a classical memory $2^{n/5}$.

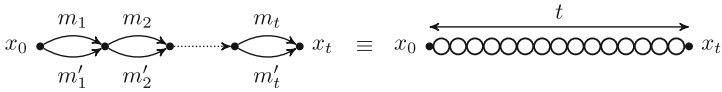


Fig. 3. Joux's multi-collision [40]

3.2 Diamond Structure and Its New Quantum Algorithm in no-QRAM setting

Kelsey and Kohno in [42] invented the diamond structure. Similar to Joux's multi-collisions and Kelsey and Schneier's expandable message [43], diamond

is also a kind of multi-collision. The difference is that, instead of mapping a single starting state to a final state in the form of sequential chain like Joux's multi-collisions, a 2^t-diamond maps a set of 2^t leaf states to a common root state as shown in Fig. 4. In classical setting, several improvements [12,44] on building diamond structure have been proposed. The time complexity to build a 2^t-diamond is $\sqrt{t} \cdot 2^{\frac{n+t}{2}}$ evaluations of the compression function of the hash function. Based on the diamond structure, Kelsey and Kohno [42] introduced the herding attack with time complexity $\sqrt{t} \cdot 2^{\frac{n+t}{2}} + 2^{n-t}$, which achieve the optimal $\mathcal{O}(2^{2n/3})$ when $t = n/3$. The memory complexity is bounded by building 2^t-diamond, which is $\mathcal{O}(2^{(n+t)/2}) = \mathcal{O}(2^{2n/3})$ [12,42].

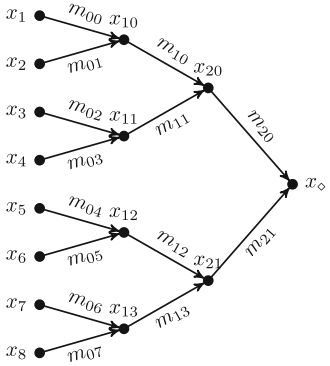


Fig. 4. 2^3-diamond [7]

Bao et al. [7] initially introduced the quantum diamond structure algorithm for both qRAM and no-qRAM scenarios. However, when we tried to replicate their algorithm, we find their no-qRAM algorithm is incorrect and for more details please refer Appendix A.

Later, at ASIACRYPT 2022, Benedikt, Fischlin, Huppert [9] presented a quantum diamond structure algorithm utilizing exponential QRACM, resulting in a time complexity of $t^{1/3} \cdot 2^{(n+2t)/3}$. Consider a level s of the 2^t-diamond structure and try to connect 2^s nodes $\{x_{s,1}, \cdots, x_{s,2^s}\}$ in a pairwise manner. Benedikt et al. split the 2^s nodes into a upper and a lower half of 2^{s-1} nodes each. For the upper half, they compute a list Y of 2^l hash evaluations $h(m_j, x_{s,i})$ with $i = 1, \cdots, 2^{s-1}$, which equally spread out over the 2^{s-1} nodes. Hence, for each node, there are $\frac{2^l}{2^{s-1}}$ hash evaluations. Store Y in QRACM, and apply Grover's algorithm to connect the first value $x_{s,2^{s-1}+1}$ of the lower half to some of these 2^l values with some message block m'. Once a connection message is found, remove the partner node from the upper half and all of its $2^l/2^{s-1}$ entries from Y. Then, add this amount of new values, again equally spread out over the remaining $2^{s-1} - 1$ values paired up, to fill the list Y up to 2^l elements again. Then connect the second node $x_{s,2^{s-1}+2}$ to Y. Continue till all 2^s nodes

are connected, then proceed with the next level $s-1$ until the entire tree is built. Benedikt et al. choose $l = \frac{n+2s}{3}$ to achieve optimal complexity to build the 2^t-diamond structure, where $s \leq t$. Therefore, the size of Y is about $2^l = 2^{\frac{n+2t}{3}}$.

To build the quantum herding attack with a 2^t-diamond structure, Benedikt et al. applied the BHT algorithm to find the M_{link} (as shown in Fig. 6). The overall complexity of the herding attack includes the complexity of building 2^t-diamond structure and finding the M_{link}, which is roughly $\mathcal{O}(2^{(n+2t)/3} + 2^{(n-t)/2})$. The optimal complexity is achieved when $t = n/7$, i.e., the optimal time complexity is $\mathcal{O}(2^{3n/7})$ with QRACM of size $\mathcal{O}(2^{(n+2t)/3}) = \mathcal{O}(2^{3n/7})$ to store Y when building 2^t-diamond structure.

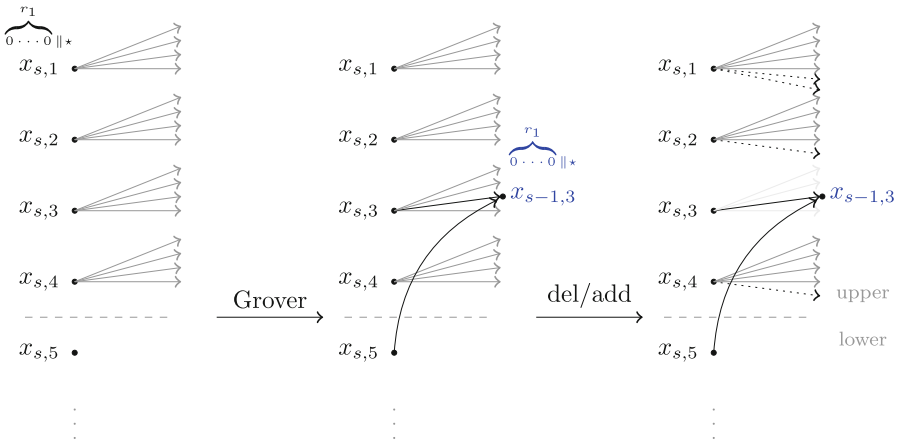


Fig. 5. Building diamond

A New no-QRAM Quantum Algorithm to Build the Diamond Structure. In this section, we introduce a quantum algorithm to build the diamond structure in no-qRAM setting based on Benedikt et al.'s [9] method and CNS collision finding algorithm [21]. As shown in Fig. 5, again consider a level s of the 2^t-diamond structure and try to connect 2^s nodes $\{x_{s,1}, \cdots, x_{s,2^s}\}$ in a pairwise manner.

1. Begin with 2^t leaf nodes that share a common suffix of r_0 0s for the purpose of connection[2].

[2] Leaf nodes with r_0 0s suffix are used for the following herding attack in Sect. 4, and are not relevant to this diamond building algorithm. After a diamond is built whose leaves are suffixed with $r_0 0$, we can apply the CNS algorithm (see Definition 2) to find a linking message whose digest collides to one of those leaves. Similar techniques for constructing distinguished points (e.g., leaves suffixed with $r_0 0$) are often used in cryptanalysis, e.g., the quantum collision or preimage finding algorithm [5,10,21],

2. Let's consider a specific level $s \leq t$ of the tree where we aim to connect the 2^s nodes $\{x_{s,1}, \ldots, x_{s,2^s}\}$ pairwise. Divide the 2^s nodes into two halves, the upper half with 2^{s-1} nodes $\{x_{s,1}, \cdots, x_{s,2^{s-1}}\}$ and the lower half with 2^{s-1} nodes $\{x_{s,2^{s-1}+1}, x_{s,2^{s-1}+2}, \cdots, x_{s,2^s}\}$. For the upper half, compute a list Y of 2^l hash values $h(m_j, x_{s,i})$ with $i = 1, \cdots, 2^{s-1}$, where the r_1 MSBs of $h(m_j, x_{s,i})$ are zero. The 2^l hash values equally spread out over the 2^{s-1} nodes, with $\frac{2^l}{2^{s-1}}$ hash values for each node. Here, similar to CNS algorithm in Sect. 2.1 to build L whose elements are prefixed with r-bit zero, we also apply Grover's algorithm to build Y. For each node $x_{s,i}$ with $i = 1, \cdots, 2^{s-1}$, run Grover's algorithm to find m_j so that the r_1 MSBs of $h(m_j, x_{s,i})$ are zero. The time to find one m_j is $2^{r_1/2}$. In order to find $\frac{2^l}{2^{s-1}}$ such m_j for node $x_{s,i}$, we apply $\frac{2^l}{2^{s-1}}$ times of Grover's algorithm. Therefore, to build Y, the total time complexity is
$$2^l \times 2^{r_1/2} = 2^{l+\frac{r_1}{2}}. \tag{8}$$

3. Store Y in a classical memory with 2^l elements $(h(m_j, x_{s,i}), m_j, x_{s,i})$ indexed by $h(m_j, x_{s,i})$. For the first node $x_{s,2^{s-1}+1}$ of the lower half, apply CNS algorithm in Sect. 2.1 to find a message block m' so that $h(m', x_{s,2^{s-1}+1})$ hits one of the entries of Y. According to Definition 2, apply $\mathrm{CNS}_h(m', Y)$ to find such m', whose time complexity is
$$\sqrt{\frac{2^{n-r_1}}{2^l}} \cdot (2^{r_1/2} + 2^l). \tag{9}$$

4. After m' is found, delete the partner node and all of its $2^l/2^{s-1}$ entries from Y. Add $2^l/2^{s-1}$ new values for Y with similar ways to Step 2 to fill Y up to 2^l elements again. Now each node of the upper half corresponds to $2^l/(2^{s-1}-1)$ elements. Delete the first node $x_{s,2^{s-1}+1}$ from lower half. The time complexity to fill Y again is
$$2^l/2^{s-1} \times 2^{r_1/2} = 2^{l-s+1+\frac{r_1}{2}}. \tag{10}$$

5. Repeat Step 3 and Step 4 until the lower half is empty. That means all the nodes of the layer of level s have been connected pairwise.

To build the layer of level s in Step 2, 2^s nodes are divided into the upper half and the lower half, each with 2^{s-1} nodes. We are going to connect all the 2^{s-1} nodes in the lower half to the upper half. In step 3, the CNS algorithm (i.e., $\mathrm{CNS}_h(m', Y)$) is applied to connect one node of the lower half (e.g., the first node $x_{s,2^{s-1}+1}$) to one node of the upper half by hitting one of the elements in Y. In Step 4, after the i-th node $x_{s,2^{s-1}+i}$ $(i = 1, \cdots, 2^{s-1}-1)$ in the lower half has been connected to Y, $\frac{2^l}{2^{s-1}-(i-1)}$ elements will be deleted from Y, and therefore, the same amount of new elements should be generated to fill up Y to

quantum k-XOR algorithm [33,53], and many classical attacks e.g. [24], to name a few. However, our Algorithm 1 is the first to apply this technique to quantum herding attack.

2^l again, whose time complexity is

$$\frac{2^l}{2^{s-1}-(i-1)} \times 2^{r_1/2}. \tag{11}$$

Since we have to connect all nodes in the lower half to the upper half, the elements in Y must be repeatedly deleted and filled up Y to 2^l for all $i = 1, 2, 3, \ldots, 2^{(s-1)} - 1$. For each i, the time to fill up Y is estimated by Eq. (11). Note that for the last node, i.e., the $i = 2^{s-1}$-th node $x_{s,2^s}$ in the lower half, we only need to apply the CNS algorithm to find a match in Y to connect the last node to the upper half, and we do not need to fill up Y again after that. Therefore, we got the component of summation in Eq. (12). Since Step 3 will be repeated for each node of the lower half, hence, $\mathrm{CNS}_h(m', Y)$ will be repeated for 2^{s-1} times. Therefore, the total time complexity to build the layer of level s is

$$T_s = 2^l \times 2^{r_1/2} + 2^{s-1} \cdot \sqrt{\frac{2^{n-r_1}}{2^l}} \cdot (2^{r_1/2} + 2^l) + \sum_{i=1}^{2^{s-1}-1} \frac{2^l}{2^{s-1}-(i-1)} \times 2^{r_1/2}. \tag{12}$$

To build the 2^t-diamond structure which includes t layers, the total time is

$$\sum_{s=t}^{1} T_s. \tag{13}$$

We could calculate

$$T_s = 2^{s-1} \cdot \sqrt{\frac{2^{n-r_1}}{2^l}} \cdot (2^{r_1/2} + 2^l) + 2^l \cdot 2^{r_1/2} \cdot \sum_{j=1}^{2^{s-1}} \frac{1}{j} = 2^{s-1} \cdot \sqrt{\frac{2^{n-r_1}}{2^l}} \cdot (2^{r_1/2} + 2^l) + O(s \cdot 2^{l+r_1/2})$$

using $\sum_{j=1}^{q} \frac{1}{j} \leq \ln q + c$ for the harmonic series. Then T_s could be minimized to $\mathcal{O}(s^{1/5} \cdot 2^{(2n+4s+4)/5})$ by setting $r_1 = 2l$ and $l = \frac{n+2s+2-2\log_2 s}{5}$.

The final complexity is obtained from summing over all t levels:

$$\begin{aligned} \sum_{s=1}^{t} \mathcal{O}(s^{1/5} \cdot 2^{(2n+4s+4)/5}) \leq & \quad \mathcal{O}(2^{(2n+4+\log_2 t)/5} \cdot \sum_{s=1}^{t} 2^{\frac{4s}{5}}) \\ = & \quad \mathcal{O}(2^{(2n+4+\log_2 t)/5} \cdot 2^{\frac{4t}{5}}) \\ = & \quad \mathcal{O}(2^{(2n+4t+4+\log_2 t)/5}), \end{aligned}$$

which is about $\mathcal{O}(2^{(2n+4t)/5})$. The classical memory is dominated by $\mathcal{O}(2^{(n+2t)/5})$ to store Y for the first layer. The number of qubits is $\mathcal{O}(n)$.

4 Herding Attack in Quantum Settings with no-QRAM

The herding attack on iterated hash function is first given by Kelsey and Kohno [42]. In the attack, the adversary chooses a public hash value h_T, and then, she

is challenged with a prefix P. Her goal is to find a suffix S such that $h_T = H(P\|S)$. At ASIACRYPT 2022, Benedikt, Fischlin, and Huppert [9] presented the quantum herding attack with $\sqrt[3]{n} \cdot 2^{3n/7}$ on iterated hash function with n-bit digest based on BHT algorithm. Their quantum attack also needs exponential qRAM inherited from the BHT algorithm [20], i.e., $2^{3n/7}$ QRACM. Therefore they left an open question on how to devise quantum herding attacks with low-qRAM. In this section, we answer the open question positively. As shown in Fig. 6, our herding attack consists in four steps:

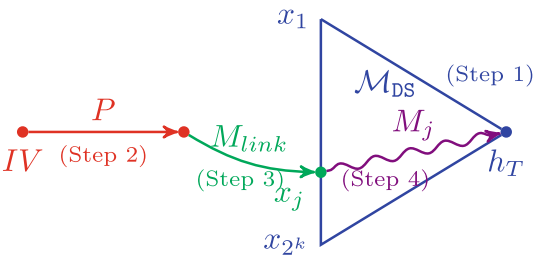


Fig. 6. Herding Attack on Iterated Hash Function [7]

- Step 1 is to build a 2^k-diamond structure. In classical herding attack by Kelsey and Kohno [42] and the quantum one by Benedikt et al. [9], the leaves x_i $(1 \le i \le 2^k)$ are randomly chosen. In our quantum attack, the r most significant bits (MSB) of x_i are zero. Store the leaves in D with classical memory
- Step 2 and Step 3 is to find a single block message M_{link} such that $h(P\|M_{link})$ collides with some value $x_j \in D$.
- Step 4 is to produce the message $M = P\|M_{link}\|M_j$, where M_j is a sequence of message blocks linking x_j to h_T with the diamond structure.

Our quantum herding attack is given in Algorithm 1.

Complexity. The time complexity to build the 2^k diamond structure is $k^{1/5} \cdot 2^{(2n+4k)/5}$ with a classical memory $k^{3/5} \cdot 2^{(n+2k)/5}$. The time complexity of the setup phase is $2^{r/2}$ with Grover algorithm. According to the quantum membership algorithm [21], the time complexity to implement $O_{f_L^h}$ is 2^k. For $(m, h(\bar{x}, m)) \in S_r^h$, $f_L^h(m) = 1$ holds with probability of $2^{k-(n-r)}$. Therefore, about $2^{\frac{n-r-k}{2}}$ calls of A, $A^\dagger$, $O_{f_L^h}$, $O_{f_L^h}^\dagger$ are needed to produce the correct $M_{link} = m$. Hence, the time complexity to find the M_{link} in Line 8 is $2^{\frac{n-r-k}{2}}(2^{r/2} + 2^k)$ with a classical memory 2^k to store L. Hence, the total time complexity is

$$2^{\frac{n-r-k}{2}}(2^{r/2} + 2^k) + k^{1/5} \cdot 2^{(2n+4k)/5}. \tag{15}$$

The classical memory complexity is bounded by the construction of the diamond structure, i.e., $k^{3/5} \cdot 2^{(n+2k)/5}$.

Algorithm 1: Herding Attack on Iterated Hash Function without qRAM

Off-line precomputation: Precompute the diamond structure using CNS quantum collision algorithm. Collect 2^k starting chaining values $D = \{x_1, x_2, \cdots, x_{2^k}\}$, where the r MSBs of $x_i \in \mathbb{F}_2^n$ are zero. The root is denoted as h_T and publish h_T.
On-line precomputation:
begin
  Receive the challenged prefix P and compute the chaining value after absorbing the message P: $\bar{x} = \bar{H}(IV, P)$.
  `/* Finding the linking message` M_{link} `by applying variant of CNS collision-finding algorithm: */`
  Store $D = \{x_1, x_2, \cdots, x_{2^k}\}$ in a classical memory L.
  Define
  $$S_r^h := \{(m, h(\bar{x}, m)) : \exists z \in \{0,1\}^{n-r}, h(\bar{x}, m) = \underbrace{0 \cdots 0}_{r \ times} \| z, z \in \{0,1\}^{n-r}\},$$
  where h is the compression function with n-bit chaining value $\bar{x}$. Let $f_L^h(m) := 1$ if $\exists x' \in L$, $h(\bar{x}, m) = x'$, and $f_L^h(m) := 0$ otherwise.
  Apply quantum amplification algorithm:
  begin
    The setup $\mathcal{A}$ is the construction of $|\phi\rangle := \frac{1}{\sqrt{|S_r^h|}} \sum_{m \in S_r^h} |m, h(\bar{x}, m)\rangle$.
    The projector is a quantum oracle query to $O_{f_L^h}$ meaning that
    $$O_{f_L^h}(|m, h(\bar{x}, m)\rangle |b\rangle) = |m, h(\bar{x}, m)\rangle |b \oplus O_{f_L^h}(m)\rangle. \tag{14}$$
  end
  Let $M_{link} = m$ and produce the message: $M = P \| M_{link} \| M_j$, where M_j is a sequence of message blocks linking x_j to h_T following the diamond structure built before.
end

The Best-Case Complexity. The optimal complexity is to balance the three formulas, i.e., $\frac{n-k}{2}$, $\frac{n-r+k}{2}$, and $\frac{2n+4k}{5}$. When $k = n/13$ and $r = 2n/13$, the optimal complexity is achieved which results in $\mathcal{O}(2^{6n/13}) = \mathcal{O}(2^{0.46n})$ time complexity and $\mathcal{O}(2^{3n/13}) = \mathcal{O}(2^{0.23n})$ classical memory.

Remark. Bao et al. [7] also proposed a no-qRAM herding attack based on a flawed method of building the diamond structure as shown in Sect. 3.2. After correcting with our right algorithm in Sect. 3.2, Bao et al.'s no-qRAM herding attack needs a time complexity of $\mathcal{O}(2^{14n/29}) = \mathcal{O}(2^{0.48n})$ with a classical memory $\mathcal{O}(2^{7n/29}) = \mathcal{O}(2^{0.24n})$, which is inferior to our attacks. For more details, please refer to Appendix B.

5 Interchange Structure and Preimage Attack on XOR Combiners

5.1 Basic Interchange Structure Technique [49]

At EUROCRYPT 2015, Leurent and Wang [49] invented the interchange structure (IS), which is used to devise a preimage attack on the XOR combiner, i.e., $\mathcal{H}_1(IV_1, M) \oplus \mathcal{H}_2(IV_2, M) = T$. The interchange structure contains a set of messages $\mathcal{M}_{\mathtt{IS}}$ and two sets of states $\mathcal{A}$ and $\mathcal{B}$, so that for any state pair $(A_i, B_j | A_i \in \mathcal{A}, B_j \in \mathcal{B})$, the attacker can pick a message $M \in \mathcal{M}_{\mathtt{IS}}$ such that $A_i = \mathcal{H}_1(IV_1, M)$ and $B_j = \mathcal{H}_2(IV_2, M)$. Suppose there is a 2^k-interchange structure (the sizes of $\mathcal{A}$ and $\mathcal{B}$ are both 2^k). In order to reach the target value T, they select a random block m, and evaluate $L_1 = \{A'_i = h_1(A_i, m), i = 1 \cdots 2^k\}$ and $L_2 = \{B'_j = T \oplus h_2(B_j, m), j = 1 \cdots 2^k\}$, where h_1 and h_2 are the compression functions. If there is a match between the two lists L_1 and L_2, then

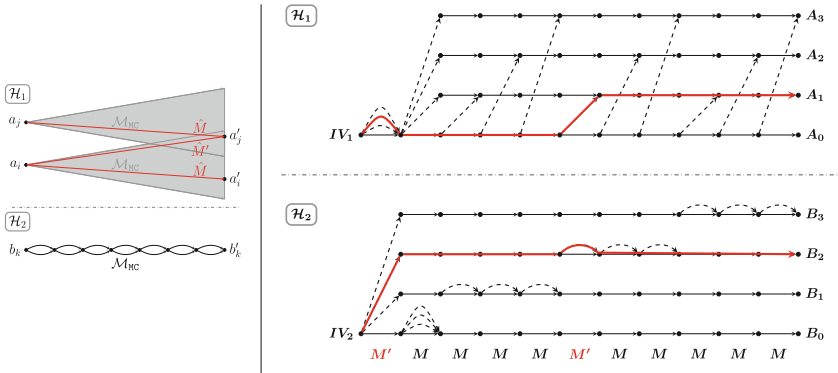
$$h_1(A_i, m) = T \oplus h_2(B_j, m) \Leftrightarrow \mathcal{H}_1(IV_1, M \| m) \oplus \mathcal{H}_2(IV_1, M \| m) = T. \quad (16)$$

The above technique is exactly a Meet-in-the-Middle approach. For a given m, it produce the preimage with probability 2^{2k-n} with time complexity 2^k. Therefore, to find the preimage, 2^{n-2k} m should be exhausted with a time complexity of $2^{n-2k} \times 2^k = 2^{n-k}$.

To build a 2^k-interchange structure (the sizes of $\mathcal{A}$ and $\mathcal{B}$ are both 2^k), the classical time complexity is $\tilde{O}(2^{2k+n/2})$ in [49].

5.2 Low qRAM Quantum Version of Interchange Structure

For the hash XOR combiners $\mathcal{H}_1(IV_1, M) \oplus \mathcal{H}_2(IV_2, M) = T$, the basic technique to build interchange structure is to build a single switch, which allows to jump from an already reachable pair of chains (a_i, b_k) to (a_j, b_k) as shown in Fig. 7(a).



(a) Building a switch

(b) Interchange structure

Fig. 7. Interchange structure and its building block [7]

Algorithm 2: Building a Single Switch in Quantum Settings with Low qRAM

1 Use the quantum Joux's multi-collision algorithm to build a set $\mathcal{M}_{\texttt{MC}}$ of 2^t messages for h_2^* that link the starting state b_k to the same state b'_k, i.e., $\forall M \in \mathcal{M}_{\texttt{MC}}, h_2^*(b_k, M) = b'_k$. The number of message blocks of M is t. Denote the i-th collision message blocks in Joux's multi-collision are (m_i^0, m_i^1), $1 \le i \le t$, which are stored in QRACM L_1, whose size is about $O(t \cdot n)$.

2 Given $|l_1, l_2, ..., l_t\rangle$ $1 \le i \le t$ and $l_i \in \{0, 1\}$, O_f is the quantum oracle that computes $O_f(|l_1, l_2, ..., l_t\rangle|0\rangle) = |l_1, l_2, ..., l_t\rangle|m_1^{l_1}, m_2^{l_2}, ..., m_t^{l_t}\rangle$ by accessing QRACM L_1. Therefore, we can obtain the superposition of Eq. (18)

a) Apply Hadamard H to the first t qubits of $|0\rangle$, we get

$$\sum_{l_1, l_2, ..., l_t \in \{0,1\}} |l_1, l_2, ..., l_t\rangle|0\rangle. \tag{17}$$

b) Apply O_f to the superposition, we get

$$|\phi\rangle = \sum_{l_1, l_2, ..., l_t \in \{0,1\}} |l_1, l_2, ..., l_t\rangle|m_1^{l_1}, m_2^{l_2}, ..., m_t^{l_t}\rangle. \tag{18}$$

3 `// the following lines are the CNS collision finding algorithm [21]`

4 Select 2^x $(x \le t)$ $M \in \mathcal{M}_{\texttt{MC}}$, where the r MSBs of $a'_j = h_1^*(a_j, M)$ are zero. Store (a'_j, M) in classical memory L_2, whose size is about 2^x. Apply Grover algorithm to produce L_2 (combining with Eq. (18)) with complexity of $2^x \cdot 2^{r/2} = 2^{x+r/2}$.

5 Let $M = (m_1^{l_1}, m_2^{l_2}, ..., m_t^{l_t}) \in \mathcal{M}_{\texttt{MC}}$, and define $g_{L_2}^{h_1^*}(M) := 1$ if $a'_i = h_1^*(a_i, M) \in L_2$, and $g_{L_2}^{h_1^*}(M) := 0$ otherwise. `// quantum membership checking`

6 Define $S_r^{h_1^*} := \{M : \exists z \in \{0,1\}^{n-r}, h_1^*(a_i, M) = \underbrace{0 \cdots 0}_{r \ times} \| z, z \in \{0,1\}^{n-r}, M \in \mathcal{M}_{\texttt{MC}}\}$.

7 Apply quantum amplification algorithm (QAA) to determine the collision.

a) The setup phase of QAA is to compute the following superposition together with Eq. (18)

$$|\phi_r\rangle := \frac{1}{\sqrt{|S_r^{h_1^*}|}} \sum_{M \in S_r^{h_1^*}} |M\rangle \tag{19}$$

b) The projector of the QAA is applying quantum oracle $O_{g_{L_2}^{h_1^*}}$, let $M = (m_1^{l_1}, m_2^{l_2}, ..., m_t^{l_t})$,

$$O_{g_{L_2}^{h_1^*}}|M\rangle|y\rangle = |M\rangle|y \oplus g_{L_2}^{h_1^*}(M)\rangle \tag{20}$$

As shown in Fig. 7(a), given the multi-collision set $\mathcal{M}_{\mathtt{MC}}$ of size 2^t, $\forall M \in \mathcal{M}_{\mathtt{MC}}$, $h_2^*(b_k, M) = b'_k$. The single switch algorithm (Alg. 2) is to find a pair $\hat{M}$, $\hat{M}' \in \mathcal{M}_{\mathtt{MC}}$, such that $h_1^*(a_j, \hat{M}) = h_1^*(a_i, \hat{M}')$.

Complexity of Algorithm 2:

- In Line 1, the time to build 2^t-$\mathcal{M}_{\mathtt{MC}}$ is $t \cdot 2^{2n/5}$, with classical memory $2^{n/5}$ by applying CNS algorithm directly.
- In Line 4, with the superposition in Eq. (18), Grover algorithm is applied to determine a $M = (m_1^{l_1}, m_2^{l_2}, ..., m_t^{l_t})$, such that the r MSBs of $h_1^*(a_j, M)$ are zero, whose time complexity is $2^{r/2}$. To find 2^x such M, the time complexity is $2^{x+r/2}$. A classical memory of size 2^x is needed to store L_2.
- In Line 7 a), the setup phase is to produce the superposition of $|\phi_r\rangle$, whose time complexity is about $2^{r/2}$.
  In Line 7 b), the projector is a quantum membership checking, whose time complexity is about 2^x. To ensure that there is at least one collision, we have $2^{t-r} \times 2^x \geq 2^{n-r}$, i.e., $t + x \geq n$. The total time complexity is

$$2^{\frac{n-r-x}{2}} \cdot (2^{r/2} + 2^x) + 2^{x+r/2} + t \cdot 2^{2n/5}. \tag{21}$$

  When $x = \frac{r}{2} = \frac{n}{5}$ and $t = \frac{4n}{5}$, we get the optimal time complexity, i.e., $\mathcal{O}(\frac{4n}{5} \cdot 2^{2n/5})$. The QRACM to store L_1 is of polynomial size, which is $\mathcal{O}(t \cdot n)$. The classical memory used to store L_2 and in Line 1 is $\mathcal{O}(2^{n/5})$.

Comparison Between Our Herding Attack and the Interchange Structure Building Algorithm. The highlevel framework for herding attack (Algorithm 1) and the interchange structure (Algorithm 2) is different, but they both apply variants of CNS collision finding algorithm [21]. As shown in Sect. 2.1 of our paper, the original full CNS algorithm is divided into two parts: the **Precomputing Part** to prepare L and the **Matching Part** to find collision with L. For our herding attack, we mainly modify the **Precomputing Part** of the original CNS to prepare the diamond whose leaf nodes are then stored in L. For interchange structure, we mainly modify the **Matching Part**. It is because different from original CNS algorithm whose messages to collide with L are chosen freely, and thus an easy Hadamard transform applied to $|0\rangle^{\otimes n}$ is enough to get the quantum superposition of the messages. However, in our attack (Algorithm 2), the messages have to be chosen from the set $\mathcal{M}_{\mathtt{MC}}$ built by Joux's multi-collision algorithm. Hence, the superposition of those messages is not trivial to obtain. To deal with it, we introduce an efficient way to build this superposition and make the attack successful.

5.3 Preimage Attack on XOR Combiners with Low qRAM

In classical setting, Leurent and Wang [49] built preimage attack on the XOR combiner with an Meet-in-the-Middle approach. Leurent and Wang first built a 2^k-interchange structure (the sizes of $\mathcal{A}$ and $\mathcal{B}$ are both 2^k) as shown in Sect. 5.1.

In this section, in quantum setting, we perform three quantum attacks on XOR combiners based on three different quantum algorithms.

Attack based on Schrottenloher-Stevens' Quantum MitM Attack [58]**.** As shown in Sect. 5.1, the sizes of L_1 and L_2 should be equal in Leurent and Wang's classical attack to achieve the optimal time complexity. However, in quantum MitM attack, according to Eq. (7), L_1 and L_2 should be of different sizes. According to (16), the matching bits are n bits, therefore, the size of $L_{\texttt{merge}}$ that contains messages satisfy (16) is very small when compared to L_1 and L_2. Actually, we only find one preimage, so that $|L_{\texttt{merge}}|$ is about 1. Without loss of generality, we assume $|L_1|$ is bigger. Then (7) is simplified as

$$|L_2| + \sqrt{|L_1|}. \tag{22}$$

To reach an optimal balance, we choose $|L_1| = 2^{2k}$ and $|L_2| = 2^k$, so that the complexity of the quantum merging algorithm is $\mathcal{O}(2^k)$. We denote this kind of interchange structure as $(2^{2k}, 2^k)$-interchange structure, which is built by applying $2^{3k} - 1$ quantum single switches (Algorithm 2) as the following:

1. Build a single switch from (a_0, b_0) to each of (a_0, b_j) $j = 0, ..., 2^k - 1$,
2. For each j, build switches from (a_0, b_j) to all (a_i, b_j) for all $i = 0, ..., 2^{2k} - 1$,
3. To reach the chain (a_i, b_j) from (a_0, b_0), we first find the switch to jump from (a_0, b_0) to (a_0, b_j) in the first step, then find the switch to jump from (a_0, b_j) to (a_i, b_j) in the second step (see Fig. 7(b)).

The time complexity is $\mathcal{O}(\frac{4n}{5} \cdot 2^{3k+2n/5})$ with $\mathcal{O}(2^{n/5})$ classical memory to build the $(2^{2k}, 2^k)$-interchange structure.

According to Lemma 1, we first guess the message block $m \in \mathbb{F}_2^g$, and compute the two lists L_1 and L_2 with $|L_1| = 2^{2k}$ and $|L_2| = 2^k$, then build the $O_{\texttt{merge}}$ with complexity $\mathcal{O}(2^k)$ according to Eq. (22). To find at least one preimage, we have $2^{g+k+2k} = 2^n$, so that $g = n - 3k$. According to Eq. (6), the time complexity of the quantum MitM attack is about $2^{\frac{n-3k}{2}} \times 2^k = 2^{\frac{n-k}{2}}$. During the quantum MitM attack, the $(2^{2k}, 2^k)$-interchange structure precomputed should be stored in QRACM, and L_2 should be stored in QRAQM. Therefore, the qRAM needed is 2^{2k}QRACM+2^kQRAQM.

The overall time complexity, including the time to build $(2^{2k}, 2^k)$-interchange structure and the quantum MitM attack, is $\frac{4n}{5} \cdot 2^{3k+2n/5} + 2^{\frac{n-k}{2}}$. The optimal complexity is $2^{17n/35} = 2^{0.485n}$ by setting $k = n/35$. The classical memory is $\mathcal{O}(2^{n/5})$. The qRAM is $2^{0.0571n}$QRACM+$2^{0.0285n}$QRAQM.

We would like to thank one of the reviewers from ASIACRYPT 2023 for pointing out an error in the preliminary version of the attack based on Schrottenloher-Stevens' method [58], and also thank him for inspiring the following two attacks.

Attack Based on Ambainis' Element Distinctness Algorithm [1]**.** To apply Ambainis' algorithm, a $(2^k, 2^k)$-interchange structure is first prepared and stored in QRACM of size about 2^k. For a guessed message block $m \in \mathbb{F}_2^g$, we build

two lists L_1 and L_2 of equal size 2^k, then apply Ambainis' quantum element distinctness algorithm to detect the collision with the time complexity of $2^{2(k+1)/3}$ and $2^{2(k+1)/3}$ QRAQM. When applying Grover's algorithm on $m \in \mathbb{F}_2^g$, the overall time complexity (including the time to build $(2^k, 2^k)$-interchange structure) to find the preimage of XOR combiner is $\frac{4n}{5} \cdot 2^{2k+2n/5} + 2^{(n-2k)/2+2(k+1)/3} \approx 2^{(2k+2n/5)} + 2^{(n/2-k/3)}$. The optimal time complexity is achieved when $k = 3n/70$, i.e., the time is $2^{(17n/35)} = 2^{(0.485n)}$, with $2^{2(k+1)/3} = 2^{0.0285n}$ QRAQM, $2^k = 2^{0.043n}$ QRACM, and $2^{n/5}$ classical memory.

Attack Based on Jaques-Schrottenloher's Golden Collision Finding Algorithm [39]**.** To apply Jaques-Schrottenloher's algorithm, a $(2^k, 2^k)$-interchange structure is first prepared and stored in QRACM of size about 2^k. In [7], Bao et al. applied Jaques and Schrottenloher's method [39] to find the collision between L_1 and L_2. Here we also apply this method in our attack. Note that Jaques and Schrottenloher found the single collision in a set of size N with $N^{6/7}$ time complexity and $N^{2/7}$ qubits, without QRAQM. Therefore, with a $(2^k, 2^k)$-interchange structure, the time complexity of our preimage attack on XOR combiner is $2^{(2k+2n/5)} + 2^{(n-2k)/2+6(k+1)/7} \approx 2^{2k+2n/5} + 2^{n/2-k/7}$ with $2^{2k/7}$ qubits, and $2^{0.2n}$ classical memory. The optimal time complexity is achieved when $k = 7n/150$, where the time complexity is $2^{37n/75} = 2^{0.493n}$, with $2^{n/75} = 2^{0.0133n}$ qubits, $2^k = 2^{0.047n}$ QRACM, and $2^{0.2n}$ classical memory.

In the no-QRAQM scenario, when compared our attack with the attack by Bao et al. [7], the time complexity is reduced from $2^{0.495n}$ to $2^{0.493n}$, and the number of qubits is significantly reduced from $2^{0.143n}$ to our $2^{0.0133n}$.

6 Collision Attack on Concatenation Combiners in Quantum Settings

For a hash concatenation combiner $\mathcal{H}_1(IV_1, M) \| \mathcal{H}_2(IV_2, M) = T_1 \| T_2$, the collision attack is to find two distinct M and M', so that $\mathcal{H}_1(IV_1, M) \| \mathcal{H}_2(IV_2, M) = \mathcal{H}_1(IV_1, M') \| \mathcal{H}_2(IV_2, M')$. Classically, based Joux's multi-collision method [40], the collision attack can be built in $\mathcal{O}(2^{n/2})$. Here, we introduce a new quantum collision attack on the hash combiners in Algorithm 3.

Complexity of Algorithm 3. Algorithm 3 is quite similar to Algorithm 2. When we let $t = n$, $x = 2^{n/5}$, $r = 2^{2n/5}$, the attack is optimal. The time complexity is $n \cdot 2^{2n/5}$ with a classical memory of $2^{n/5}$ and polynomial number of qubits.

Algorithm 3: Collision attack on Concatenation combiners in Quantum Settings with Low qRAM

1 Use the quantum Joux's multi-collision algorithm to build a set $\mathcal{M}_{\texttt{MC}}$ of 2^t messages for $\mathcal{H}_2$ that link the starting state IV_2 to the same state T_2, i.e., $\forall M \in \mathcal{M}_{\texttt{MC}}, \mathcal{H}_2(IV_2, M) = T_2$. The block length of M is t. Denote the i-th collision message blocks in Joux's multi-collision are (m_i^0, m_i^1), $1 \leq i \leq t$. Store (m_i^0, m_i^1) in QRACM L_1 (to be used in the construction of superposition), whose size is about $O(t \cdot n)$.

2 Given $|l_1, l_2, ..., l_t\rangle$ $1 \leq i \leq t$ and $l_i \in \{0, 1\}$, O_f is the quantum oracle that computes $O_f(|l_1, l_2, ..., l_t\rangle|0\rangle) = |l_1, l_2, ..., l_t\rangle|m_1^{l_1}, m_2^{l_2}, ..., m_t^{l_t}\rangle$ by accessing QRACM L_1. Therefore, we can obtain the superposition of Eq. (24)

a) Apply Hadamard H to the first t qubits of $|0\rangle$, we get

$$\sum_{l_1, l_2, ..., l_t \in \{0,1\}} |l_1, l_2, ..., l_t\rangle|0\rangle. \tag{23}$$

b) Apply O_f to the superposition, we get

$$|\phi\rangle = \sum_{l_1, l_2, ..., l_t \in \{0,1\}} |l_1, l_2, ..., l_t\rangle|m_1^{l_1}, m_2^{l_2}, ..., m_t^{l_t}\rangle. \tag{24}$$

3 Select 2^x $(x \leq t)$ $M \in \mathcal{M}_{\texttt{MC}}$, where the r MSBs of $T_1 = \mathcal{H}_1(IV_1, M)$ are zero. Store (T_1, M) in classical memory L_2, whose size is about 2^x. L_2 is produced by applying Grover algorithm and combining with Eq. (24). The time complexity is $2^x \cdot 2^{r/2} = 2^{x+r/2}$.

4 Let $M = (m_1^{l_1}, m_2^{l_2}, ..., m_t^{l_t}) \in \mathcal{M}_{\texttt{MC}}$, and define $g_{L_2}^{\mathcal{H}_1}(M) := 1$ if $y = \mathcal{H}_1(IV_1, M) \in L_2$, and $g_{L_2}^{\mathcal{H}_1}(M) := 0$ otherwise. `/* The quantum membership algorithm. */`

5 Define $S_r^{\mathcal{H}_1} := \{M : \exists z \in \{0,1\}^{n-r}, \mathcal{H}_1(IV_1, M) = \underbrace{0 \cdots 0}_{r \ times} \|z, z \in \{0,1\}^{n-r}, M \in \mathcal{M}_{\texttt{MC}}\}$.

6 `/* Run a variant of CNS algorithm. Apply quantum amplification algorithm (QAA). */`

7 The setup phase of QAA is the construction

$$|\phi_r\rangle := \frac{1}{\sqrt{|S_r^{\mathcal{H}_1}|}} \sum_{x \in S_r^{\mathcal{H}_1}} |M\rangle \tag{25}$$

8 The projector of the QAA is applying quantum oracle $O_{g_{L_2}^{\mathcal{H}_1}}$, let $M = (m_1^{l_1}, m_2^{l_2}, ..., m_t^{l_t})$,

$$O_{g_{L_2}^{\mathcal{H}_1}}|M\rangle|y\rangle = |M\rangle|y \oplus g_{L_2}^{\mathcal{H}_1}(M)\rangle \tag{26}$$

7 Herding Attack on Concatenation Combiners in Quantum Setting

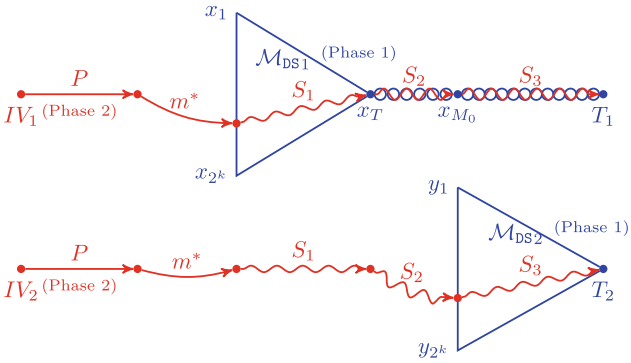


Fig. 8. Herding Attack on Concatenation Combiners in quantum settings [7]

The herding attack on concatenation combiners in quantum settings is given in Fig. 8 and Algorithm 4. In the off-line precomputation phase (Line 2 to 6), two diamond structures $\mathcal{M}_{\mathtt{DS1}}$ and $\mathcal{M}_{\mathtt{DS2}}$ are built, two Joux's multi-collisions $\mathcal{M}_{\mathtt{MC}_s}$ and $\mathcal{M}_{\mathtt{MC}_\ell}$ are built. The root node of $\mathcal{M}_{\mathtt{DS1}}$ is x_T. The 2^t-Joux's multi-collision $\mathcal{M}_{\mathtt{MC}_s}$ links x_T to x_{M_0}. Then the $2^{k\cdot\frac{4n}{5}}$-Joux's multi-collision $\mathcal{M}_{\mathtt{MC}_\ell}$ is built to link x_{M_0} and T_1. After that, 2^k-diamond $\mathcal{M}_{\mathtt{DS2}}$ is built. Here, we only apply the CNS collision finding algorithm to build the diamond without using the method given in Sect. 3.2. This is because, in the diamond building algorithm given in Sect. 3.2, one has to frequently apply Grover's algorithm to find message blocks to fill up Y to 2^l. For herding attack on MD hash (Algorithm 1), the message blocks are freely chosen. Therefore, the superposition of the message for Grover's algorithm is easy to generate, and a trivial Hadamard transformation on $|0\rangle^{\otimes n}$ is enough. However, when building the 2^k-diamond $\mathcal{M}_{\mathtt{DS2}}$, those message blocks have to be selected from a prefixed message set constructed by Joux's multi-collision algorithm, i.e., $\mathcal{M}_{\mathtt{MC}_\ell}$. To frequently build the superposition of messages from $\mathcal{M}_{\mathtt{MC}_\ell}$ for applying Grover's algorithm is not easy. Therefore, we use the trivial method that only uses CNS collision finding algorithm here. Since there are $2^k - 1$ collisions in a 2^k-diamond, $2^k - 1$ times of CNS algorithm are needed to build the diamond. To build the 2^k-diamond $\mathcal{M}_{\mathtt{DS1}}$, we can freely use diamond building algorithm in Sect. 3.2 or apply CNS algorithm trivially, since the time to build $\mathcal{M}_{\mathtt{MC}_\ell}$ already bound the complexity, we just choose CNS algorithm trivially to build $\mathcal{M}_{\mathtt{DS1}}$ without increasing the overall complexity. Similar reason also prevents us from applying diamond building algorithm given in Sect. 3.2 to Algorithm 5.

Algorithm 4: Quantum Herding Attack on Concatenation Combiners with low qRAM

Off-line precomputation:
begin
Build a diamond $\mathcal{M}_{\texttt{DS1}}$ for $\mathcal{H}_1$, which starts from 2^k states $D_1 = \{x_i\}_1^{2^k}$, where the r MSBs of $x_i \in \mathbb{F}_2^n$ are zero. To build $\mathcal{M}_{\texttt{DS1}}$, we do not use the method given in Section 3.2, but only use CNS algorithm to build each collision until the root x_T is derived. Totally, $2^{k-1} + 2^{k-2} + \cdots + 1 = 2^k - 1$ times of CNS are applied with time complexity $2^{k+2n/5}$ and memory complexity of $2^{n/5}$. The root is x_T. From the hash value x_T, build a 2^t-Joux's multi-collision $\mathcal{M}_{\texttt{MC}_s}$, in which all messages map x_T to a state x_{M_0}. Continue to build a $2^{k \cdot \frac{4n}{5}}$-Joux's multi-collision $\mathcal{M}_{\texttt{MC}_\ell}$ (consists of k fragments and each fragment is of length $4n/5$) on $\mathcal{H}_1$ from the starting state x_{M_0} and mapping to the state T_1. Denote the terminal states of each of the k fragments of $\mathcal{M}_{\texttt{MC}_\ell}$ by x_{M_i} for i from 1 to k (note that $x_{M_k} = T_1$).
Build a diamond $\mathcal{M}_{\texttt{DS2}}$ for $\mathcal{H}_2$, which starts from 2^k states $D_2 = \{y_i\}_1^{2^k}$, where the r MSBs of $y_i \in \mathbb{F}_2^n$ are zero.. The messages used to building $\mathcal{M}_{\texttt{DS2}}$ are all chosen from the set $\mathcal{M}_{\texttt{MC}_\ell}$. For example, the messages mapping the first layer of 2^k states to the 2^{k-1} states in $\mathcal{M}_{\texttt{DS2}}$ are chosen from the set of $2^{4n/5}$ messages in the first fragment of $\mathcal{M}_{\texttt{MC}_\ell}$ mapping x_{M_0} to x_{M_1}. To build $\mathcal{M}_{\texttt{DS2}}$, we do not use the method given in Section 3.2, but only apply $2^k - 1$ times CNS algorithm variant given by Algorithm 2 to find $2^k - 1$ collisions in $\mathcal{M}_{\texttt{MC}_\ell}$. Note that Algorithm 2 is exactly the method to find two messages from a set of multi-collisions that make two states collides (as shown in Figure 7(a)). The time to build $\mathcal{M}_{\texttt{DS2}}$ is $O(2^{k+2n/5})$ with a classical memory $2^{n/5}$.
Commit $T_1 \| T_2$ to the public.
end
On-line phase:
begin
Receive the challenged prefix P and compute the internal chaining value $x_P = h_1^*(IV_1, P)$ and $y_P = h_2^*(IV_2, P)$.
`/* Finding the linking message` m^* `by applying variant of CNS collision-finding algorithm: */`
Store D_1 in a classical memory L_1.
Apply Line 6 to 12 of Algorithm 1 to determine linking message m^* that maps x_P to one of the leaf state x_j of $\mathcal{M}_{\texttt{DS1}}$, and retrieve the message S_1 that link the leaf x_j to the root x_T.
Compute $y_T = h_2^*(IV_2, P \| m^* \| S_1)$.
`/* Finding the linking message` S_2 `by applying variant of CNS collision-finding algorithm: */`
Store D_2 in a classical memory L_2.
Apply CNS algorithm variant given by Algorithm 2 to find $S_2 \in \mathcal{M}_{\texttt{MC}_s}$, which maps y_T to one of the leaf state y_j of $\mathcal{M}_{\texttt{DS2}}$, and retrieve the message S_3 that link the leaf y_j to the root T_2.
$M = P \| m^* \| S_1 \| S_2 \| S_3$ is the returned message.
end

Algorithm 5: Quantum Herding attack on Zipper Hash with Low qRAM

Off-line phase: begin
  Build a $2^{k\cdot\frac{4n}{5}}$-Joux's multi-collision $\mathcal{M}_{\texttt{MC1}}$ (consists of k fragments and each fragment is of length $4n/5$) that link IV and x_{M_0}. Denote the terminal states of each of the k fragments of $\mathcal{M}_{\texttt{MC1}}$ by x_{M_i} for i from $k-1$ to 0.
  Build 2^t-Joux's multi-collision $\mathcal{M}_{\texttt{MC2}}$ from x_{M_0} to $\bar{x}$.
  Build $\mathcal{M}_{\texttt{DS}}$, which starts from 2^k leaf states $D = \{y_i\}_1^{2^k}$ to the root state h_T, where the r MSBs of $y_i \in \mathbb{F}_2^n$ are zero. Similar to Line 4 of Algorithm 4, we apply $2^k - 1$ times of Algorithm 2 to build $\mathcal{M}_{\texttt{DS}}$, which needs $2^{k+2n/5}$ time and $2^{n/5}$ memory.
  Commit h_T.
end
On-line phase: begin
  Given the suffix S, compute $\bar{y} = h_2^*(h_1^*(\bar{x}, S), \overleftarrow{S})$.
  Apply the variant of CNS to find the $m^* \in \mathcal{M}_{\texttt{MC2}}$ to connect $\bar{y}$ with the y_j one of the leaf states of $\mathcal{M}_{\texttt{DS}}$, and retrieve the corresponding message $S_1 \in \mathcal{M}_{\texttt{MC2}}$.
  Output the message $S_1 \| m^* \| S$.
end

Complexity of Algorithm 4.

- In the off-line precompuation phase (Line 2 to 6), the time complexity to build $\mathcal{M}_{\texttt{DS1}}$, $\mathcal{M}_{\texttt{MC}_s}$, $\mathcal{M}_{\texttt{MC}_\ell}$, and $\mathcal{M}_{\texttt{DS2}}$ is

$$2^{k+2n/5} + t \cdot 2^{2n/5} + 4nk/5 \cdot 2^{2n/5} + 2^{k+2n/5} \approx 2^{k+2n/5},$$

  where $t = \mathcal{O}(n)$.
- In the online phase (Line 8 to 18), the time to find m^* and S_2 are both $2^{\frac{n-r-k}{2}}(2^{r/2} + 2^k)$.

Therefore, the overall optimal time complexity of Algorithm 4 is $\mathcal{O}(2^{7n/15})$ by balancing the off-line and on-line computation phases and assigning $k = n/15$, $r = 2k$, and $t = n$. The memory cost is dominated by building Joux's multi-collision with CNS, i.e., $\mathcal{O}(2^{n/5})$ classical memory.

8 Quantum Herding Attack on Hash-Twice

The attack on Hash-Twice shares the fundamental ideas of the attack on the concatenation combiners, as depicted in Fig. 9. The attacker selects T_2 as their commitment and subsequently faces a challenge involving an unknown prefix P. The attack is the same to the attack on concatenation combiner. Please see Algorithm 4 for details. The only difference is that the IV_2 is replaced by T_1. Therefore, the overall optimal time complexity is also $\mathcal{O}(2^{7n/15})$ with a classical memory of $\mathcal{O}(2^{n/5})$.

9 Quantum Herding Attack on Zipper Hash

As stated by Andreeva et al. [3], the traditional herding attack with a prefix P can not be applied to Zipper Hash. Therefore, Andreeva et al. [3] gave a variant of the herding attack, where the challenge is placed at the end: as shown in Fig. 10, the adversary commits to a hash value h_T, then she is challenged with a suffix S, and has to produce $S_1 \| m^*$ such that $\mathcal{H}(IV, S_1 \| m^* \| S) = h_T$. The complexity of Andreeva et al.'s classical attack is $\mathcal{O}(2^{2n/3})$.

In this section, we introduce a quantum version Andreeva et al.'s attack in Algorithm 5. The complexity of the off-line phase dominated by building $\mathcal{M}_{\mathtt{DS}}$, which is about $\mathcal{O}(2^{k+2n/5})$. The on-line phase is $2^{\frac{n-r-k}{2}} \cdot (2^{r/2} + 2^k)$ with $t = n$. Let $k = \frac{n}{15}$, $r = 2k$, the optimal complexity is achieved to be $2^{7n/15}$. The memory is $2^{n/5}$.

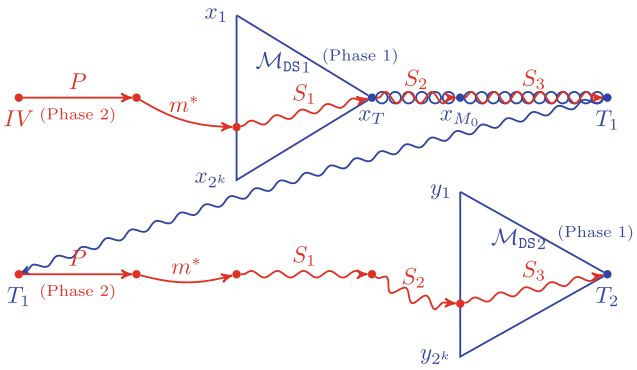


Fig. 9. Herding attack on Hash-Twice

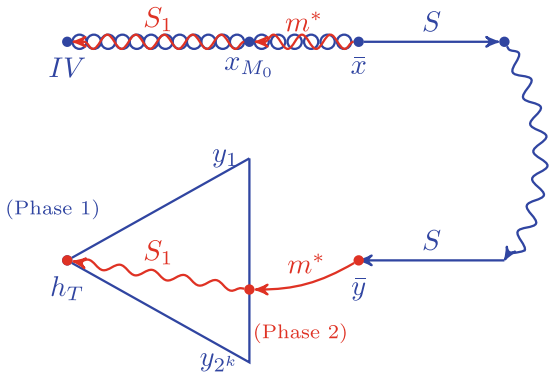


Fig. 10. Herding attack on Zipper Hash

Conclusion

This paper evaluated the quantum attacks on iterated hash functions and various important hash combiners. Most of the attacks do not need qRAM anymore, and the quantum preimage attack on hash XOR combiner is improved by significantly reducing the number of qubits from previous $2^{0.143n}$ to the current $2^{0.013n}$. Since the existence of large qRAM is still questionable, building quantum attacks with low-qRAM is of more practical relevance. Since for hash functions, the attackers do not need online superposition queries, quantum attacks on hash functions are more friendly than on other keyed primitives like block ciphers. Therefore, exploring the quantum attacks on hash functions is of more practical relevance.

Acknowledgements. We thank the anonymous reviewers of ASIACRYPT 2023 for their insightful comments, and a special thanks to our shepherd for providing so much wonderful guidance and co-inventing some algorithms that greatly improved the paper. This work is supported by the National Key R&D Program of China (2022YFB2702804, 2018YFA0704701), the Natural Science Foundation of China (62272257, 62302250, 62072270, 62072207), Shandong Key Research and Development Program (2020ZLYS09), the Major Scientific and Technological Innovation Project of Shandong, China (2019JZZY010133), the Major Program of Guangdong Basic and Applied Research (2019B030302008, 2022A1515140090), Key Research Project of Zhejiang Province, China (2023C01025).

A On Bao et al.'s Diamond Structure Building Algorithm

In [7, Section 3.3], the authors proposed a quantum algorithm for building diamond structure with exponential large QRAQM in their Algorithm 2. After that, they try to study the no-qRAM version. However, they only give the following sentences for their no-qRAM algorithm [7, Page 12]:

> "*In Scenario R2, the time complexity to find a collision is of* $(2^{(n-t)})^{2/5}$ *computations. Therefore, building a* 2^t*-diamond structure requires* $\mathcal{O}(t^{(2/3)} \cdot 2^t \cdot 2^{(2(n-t)/5)}) = \mathcal{O}(t^{2/3} \cdot 2^{(2n+3t)/5})$ *computations, with* $\mathcal{O}(t^{2/3} \cdot 2^t \cdot 2^{(n-t)/5}) = \mathcal{O}(t^{2/3} \cdot 2^{(n+4t)/5})$ *classical memory.* (see [7, Page 12])"

The authors do not give concrete steps for this no-qRAM algorithm. After communicating with the authors, we know that they just estimated the time complexity by replacing the Grover's algorithm with CNS algorithm [21] and use classical memory to store the data instead of qRAM. They do not give the concrete steps at all.

However, the conversion is not trivial as estimated by the authors of [7]. In fact, we use almost two pages in Sect. 3.2 to reveal the no-qRAM algorithm. When we try to rebuild the steps with CNS collision algorithm [21] for building diamond, we find the final time complexity is $2^{(2n+4t)/5}$, which is different from the time $2^{(2n+3t)/5}$ claimed in [7]. Then, we communicated with the authors of [7] again, and they admitted our steps and time evaluation are correct.

Since the authors of [7] do not publish or give us their concrete steps for their claimed no-qRAM algorithm, we can not check which step is possibly wrong or which step leads to the different complexities. Since the herding attack is based on diamond structure, Bao et al's [7] herding attack in no-qRAM setting is also flawed.

B On Bao et al.'s Quantum Herding Attack

In the original estimation by Bao et al. [7, Section 4.3], the overall time complexity of the no-qRAM herding attack is about $2^{((2n+3k)/5)} + 2^{(n/2-k/6)}$, where $2^{((2n+3k)/5)}$ is the time to build a 2^k-diamond, and the time $2^{(n/2-k/6)}$ is to find the linking message M_{link} to the diamond based on Chailloux et al.'s multi-target preimage algorithm [21]. After tradeoff between the two, it achieves optimal when $k = 3n/23$, which results in the overall time complexity $2^{(11n/23)} = 2^{(0.478n)}$, classical memory $2^{(7n/23)} = 2^{(0.304n)}$. Even if we compare our no-qRAM herding attack in Sect. 4 (i.e., time $2^{0.46n}$, classical memory $2^{0.23n}$) with this original complexity estimation of [7], the improvement of our attack is obvious.

However, the algorithm of building diamond structure of [7] is flawed as shown in Appendix A. Their herding attack based on diamond is also wrong. In fact, the time $2^{((2n+3k)/5)}$ will be $2^{((2n+4k)/5)}$ when using our correct diamond building algorithm given in Sect. 3.2. Therefore, the complexity of Bao et al.'s no-qRAM herding attack becomes $2^{((2n+4k)/5)} + 2^{(n/2-k/6)}$ time and $2^{(n+2k)/5}$ classical memory, which achieves optimal when $k = 3n/29$, that results in time $2^{(14n/29)} = 2^{(0.4827n)}$, classical memory $2^{(7n/29)} = 2^{(0.24n)}$. When compared with this corrected Bao et al.'s attack, our attack in Sect. 4 (time=$2^{0.46n}$, classical memory=$2^{0.23n}$) is still better obviously.

References

1. Ambainis, A.: Quantum walk algorithm for element distinctness. SIAM J. Comput. **37**(1), 210–239 (2007)
2. Andreeva, E., et al.: New second-preimage attacks on hash functions. J. Cryptol. **29**(4), 657–696 (2016)
3. Andreeva, E., Bouillaguet, C., Dunkelman, O., Kelsey, J.: Herding, second preimage and trojan message attacks beyond Merkle-Damgård. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 393–414. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05445-7_25
4. Andreeva, E., et al.: Second preimage attacks on dithered hash functions. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 270–288. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_16
5. Banegas, G., Bernstein, D.J.: Low-communication parallel quantum multi-target preimage search. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 325–335. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72565-9_16
6. Bao, Z., Dinur, I., Guo, J., Leurent, G., Wang, L.: Generic attacks on hash combiners. J. Cryptol. **33**(3), 742–823 (2020)

7. Bao, Z., Guo, J., Li, S., Pham, P.: Evaluating the security of Merkle-Damgård hash functions and combiners in quantum settings. In: Yuan, X., Bai, G., Alcaraz, C., Majumdar, S. (eds.) NSS 2022. LNCS, vol. 13787, pp. 687–711. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-23020-2_39
8. Bao, Z., Wang, L., Guo, J., Gu, D.: Functional graph revisited: updates on (second) preimage attacks on hash combiners. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 404–427. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_14
9. Benedikt, B.J., Fischlin, M., Huppert, M.: Nostradamus goes quantum. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022. LNCS, vol. 13793, pp. 583–613. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22969-5_20
10. Bernstein, D.J.: Cost analysis of hash collisions: will quantum computers make SHARCS obsolete. SHARCS **9**, 105 (2009)
11. Biham, E., Dunkelman, O.: A framework for iterative hash functions - HAIFA. IACR Cryptology ePrint Archive, p. 278 (2007)
12. Blackburn, S.R., Stinson, D.R., Upadhyay, J.: On the complexity of the herding attack and some related attacks on hash functions. Des. Codes Cryptogr. **64**(1–2), 171–193 (2012)
13. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Yu., Schrottenloher, A.: Quantum attacks without superposition queries: the offline Simon's algorithm. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 552–583. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_20
14. Bonnetain, X., Jaques, S.: Quantum period finding against symmetric primitives in practice. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(1), 1–27 (2022)
15. Bonnetain, X., Leurent, G., Naya-Plasencia, M., Schrottenloher, A.: Quantum linearization attacks. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 422–452. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92062-3_15
16. Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part I. LNCS, vol. 11272, pp. 560–592. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03326-2_19
17. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: On quantum slide attacks. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019. LNCS, vol. 11959, pp. 492–519. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38471-5_20
18. Bonnetain, X., Schrottenloher, A., Sibleyras, F.: Beyond quadratic speedups in quantum attacks on symmetric schemes. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 315–344. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-07082-2_12
19. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemp. Math. **305**, 53–74 (2002)
20. Brassard, G., HØyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 163–169. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054319
21. Chailloux, A., Naya-Plasencia, M., Schrottenloher, A.: An efficient quantum collision search algorithm and implications on symmetric cryptography. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 211–240. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_8
22. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_39

23. Dierks, T., Allen, C.: The TLS protocol version 1.0. Technical report (1999)
24. Dinur, I.: New attacks on the concatenation and XOR hash combiners. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 484–508. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_19
25. Dong, X., Dong, B., Wang, X.: Quantum attacks on some feistel block ciphers. Des. Codes Cryptogr. **88**(6), 1179–1203 (2020)
26. Dong, X., Guo, J., Li, S., Pham, P.: Triangulating rebound attack on AES-like hashing. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. LNCS, vol. 13507, pp. 94–124. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15802-5_4
27. Dong, X., Sun, S., Shi, D., Gao, F., Wang, X., Hu, L.: Quantum collision attacks on AES-like hashing with low quantum random access memories. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 727–757. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_25
28. Dong, X., Zhang, Z., Sun, S., Wei, C., Wang, X., Hu, L.: Automatic classical and quantum rebound attacks on AES-like hashing by exploiting related-key differentials. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 241–271. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92062-3_9
29. Flórez Gutiérrez, A., Leurent, G., Naya-Plasencia, M., Perrin, L., Schrottenloher, A., Sibleyras, F.: New results on Gimli: full-permutation distinguishers and improved collisions. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 33–63. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_2
30. Freier, A., Karlton, P., Kocher, P.: The secure sockets layer (SSL) protocol version 3.0. Technical report (2011)
31. Giovannetti, V., Lloyd, S., Maccone, L.: Architectures for a quantum random access memory. Phys. Rev. A **78**(5), 052310 (2008)
32. Giovannetti, V., Lloyd, S., Maccone, L.: Quantum random access memory. Phys. Rev. Lett. **100**(16), 160501 (2008)
33. Grassi, L., Naya-Plasencia, M., Schrottenloher, A.: Quantum algorithms for the k-xor problem. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part I. LNCS, vol. 11272, pp. 527–559. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03326-2_18
34. Grover, L.K.: 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, 22–24 May 1996, pp. 212–219 (1996)
35. Hosoyamada, A., Sasaki, Yu.: Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 198–218. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76953-0_11
36. Hosoyamada, A., Sasaki, Yu.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. IACR Cryptology ePrint Archive 2020:213 (2020)
37. Hosoyamada, A., Sasaki, Yu.: Quantum collision attacks on reduced SHA-256 and SHA-512. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 616–646. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84242-0_22
38. Ito, G., Hosoyamada, A., Matsumoto, R., Sasaki, Yu., Iwata, T.: Quantum chosen-ciphertext attacks against feistel ciphers. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 391–411. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12612-4_20

39. Jaques, S., Schrottenloher, A.: Low-gate quantum golden collision finding. In: Dunkelman, O., Jacobson, Jr., M.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 329–359. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81652-0_13
40. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_19
41. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 207–237. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_8
42. Kelsey, J., Kohno, T.: Herding hash functions and the nostradamus attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 183–200. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_12
43. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_28
44. Kortelainen, T., Kortelainen, J.: On diamond structures and trojan message attacks. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 524–539. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_27
45. Kuperberg, G.: Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In: Severini, S., Brandão, F.G.S.L. (eds.) 8th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2013, 21–23 May 2013, Guelph, Canada, volume 22 of LIPIcs, pp. 20–34. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2013)
46. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: IEEE International Symposium on Information Theory, ISIT 2010, 13–18 June 2010, Austin, Texas, USA, Proceedings, pp. 2682–2685 (2010)
47. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012, Honolulu, HI, USA, 28–31 October 2012, pp. 312–316 (2012)
48. Leander, G., May, A.: Grover meets Simon – quantumly attacking the FX-construction. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 161–178. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_6
49. Leurent, G., Wang, L.: The sum can be weaker than each part. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 345–367. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_14
50. Liskov, M.: Constructing an ideal hash function from weak ideal compression functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74462-7_25
51. Mendel, F., Rechberger, C., Schläffer, M.: MD5 is weaker than weak: attacks on concatenated combiners. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 144–161. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_9
52. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21

53. Naya-Plasencia, M., Schrottenloher, A.: Optimal merging in quantum k-xor and k-sum algorithms. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 311–340. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_11
54. Nielsen, Chuang, I.L.: Quantum Computation and Quantum Information, 10th Anniversary edn. Cambridge University Press, Cambridge (2016)
55. NIST. The post quantum project. https://csrc.nist.gov/projects/post-quantum-cryptography
56. Preneel, B.: Analysis and design of cryptographic hash functions. Ph.D. thesis, Katholieke Universiteit te Leuven Leuven (1993)
57. Schrottenloher, A.: Quantum linear key-recovery attacks using the QFT. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023. LNCS, vol. 14085, pp. 258–291. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38554-4_9
58. Schrottenloher, A., Stevens, M.: Simplified MITM modeling for permutations: new (quantum) attacks. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 717–747. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15982-4_24
59. Shor, P.W.: 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, pp. 124–134 (1994)
60. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**(5), 1474–1483 (1997)
61. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 570–596. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_19
62. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_2
63. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_2