IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2021, No. 1, pp. 316–336. DOI:10.46586/tosc.v2021.i1.316-336

Quantum Free-Start Collision Attacks on Double Block Length Hashing with Round-Reduced AES-256

Amit Kumar Chauhan[1]✉, Abhishek Kumar[1] and Somitra Kumar Sanadhya[2]

[1] Indian Institute of Technology, Ropar, India
{2017csz0008,2017csz0009}@iitrpr.ac.in
[2] Indian Institute of Technology, Jodhpur, India
somitra@iitj.ac.in

Abstract. Recently, Hosoyamada and Sasaki (EUROCRYPT 2020), and Xiaoyang Dong et al. (ASIACRYPT 2020) proposed quantum collision attacks against AES-like hashing modes AES-MMO and AES-MP. Their collision attacks are based on the quantum version of the rebound attack technique exploiting the differential trails whose probabilities are too low to be useful in the classical setting but large enough in the quantum setting. In this work, we present dedicated quantum free-start collision attacks on Hirose's double block length compression function instantiated with AES-256, namely HCF-AES-256. The best publicly known classical attack against HCF-AES-256 covers up to 9 out of 14 rounds. We present a new 10-round differential trail for HCF-AES-256 with probability 2^{-160}, and use it to find collisions with a quantum version of the rebound attack. Our attack succeeds with a time complexity of $2^{85.11}$ and requires 2^{16} qRAM in the quantum-attack setting, where an attacker can make only classical queries to the oracle and perform offline computations. We also present a quantum free-start collision attack on HCF-AES-256 with a time complexity of $2^{86.07}$ which outperforms Chailloux, Naya-Plasencia, and Schrottenloher's generic quantum collision attack (ASIACRYPT 2017) in a model when large qRAM is not available.

Keywords: Quantum collision attacks · Rebound attacks · qRAM · AES-256 · Double block length compression function.

1 Introduction

The prospect of large scale quantum computers have prompted scrutiny towards the post-quantum security analysis of cryptographic primitives. In public-key setting, Shor's seminal work [Sho94] for factoring integers and computing discrete logarithms will break public-key schemes such as RSA, ECDSA, and ECDH in polynomial time. In symmetric-key setting, it was generally believed that Grover's algorithm [Gro96] would provide the quadratic speedup in exhaustive search attack against the symmetric-key schemes such as block ciphers and hash functions, and thus doubling the key length addresses the concern. Interestingly, this belief has now been challenged due to several dedicated quantum attacks, such as on block ciphers [BNS19b], hash functions [HS20, DSS+20], message authentication codes, authenticated encryption schemes [KM10, KLLN16, Bon17, LM17, HS18, BNS19a, IHM+19, DDW20] etc. These attacks primarily rely on Simon's algorithm [Sim97] requiring online quantum superposition queries, except in [BHN+19] where offline queries are performed. However, the practical relevance of making online quantum superposition queries to the keyed primitives is controversial.

Received: 2020-11-23 Accepted: 2021-01-23 Published: 2021-03-19

In contrast, finding collisions for hash functions does not require any online quantum superposition queries since all computations are public and can be performed offline. In the classical setting, the generic attack complexity to find collisions against an n-bit hash function is $O(2^{n/2})$ using the birthday paradox. In the quantum setting, BHT algorithm [BHT98] finds collisions with a query complexity of $O(2^{n/3})$ provided that $O(2^{n/3})$-qubit quantum random access memory (qRAM) is available. Therefore, any dedicated attack with less than $O(2^{n/3})$ quantum complexity will be regarded as a meaningful attack. However, given the current state of development of quantum computers, it is generally admitted that large qRAMs are not feasible in the near future. Therefore quantum algorithms using small or no qRAMs are preferable. Chailloux, Naya-Plasencia, and Schrottenloher [CNS17] present a collision finding algorithm with a query complexity of $O(2^{2n/5})$, a classical memory of $O(2^{n/5})$, and only $O(n)$ quantum memory. This is the only algorithm in the literature that beats the classical birthday bound without using a large qRAM.

In this work, we target to attack double block-length hash functions in the quantum setting. The double block length (DBL) hashing is a well-established method of constructing a compression function with $2n$-bit output based only on an n-bit block cipher. DBL hash functions have an obvious advantage over classical block cipher based functions such as the PGV hash modes [PGV93, BRSS10] in that the same type of underlying primitive allows for a larger compression function. The original idea dates back to the designs of MDC-2 and MDC-4 in 1988 by Meyer and Schilling [MS88]. Since then, many schemes have been presented following this approach [Mer89, LM92, HLMW93, Hir04, Hir06, Sta08, Sta09, FGL09, AFK+11, Men17]. In particular, Hirose suggested a more efficient DBL construction by using two different provably secure block ciphers inside the compression function. Armknecht et al. [AFK+11] showed preimage resistance and collision resistance of Hirose [Hir06], Abreast-DM [LM92] and Tandem-DM [FGL09] compression functions.

In 2009, Mendel et al. [MRST09] introduced the rebound attack as a variant of differential cryptanalysis and applied it to the hash function Whirlpool, standardized by ISO/IEC. Lambereger et al. [LMS+15] further improved the rebound attacks by introducing multiple inbound phases. Following rebound attack techniques, Chen et al. [CHKM14] proposed the first free-start collision attack on DBL compression function when the underlying block cipher is instantiated with AES-256. Their attacks work for 6, 8, and 9 rounds of the construction with time complexity of 2^8, 2^{96}, and 2^{120}, respectively in the classical setting. Recently, Hosoyamada and Sasaki [HS20] presented a dedicated quantum collision attack on 7-round AES-MMO and 6-round Whirlpool when a large qRAM is available. Later, Xiaoyang Dong et al. [DSS+20] presented an improved quantum version of rebound attacks on 7-round AES-MMO and 5-round Grøstel-512 in the setting where a small qRAM is available. Their collision attacks use the rebound attack technique to exploit the differential trails, which have probabilities too low to be useful in the classical setting but large enough in the quantum setting. Motivated by their works, we apply a quantum version of rebound attacks with multiple inbound phases to find collisions on Hirose's DBL compression function when we instantiate the underlying block cipher with AES-256.

1.1 Our Contribution

This paper describes the first dedicated quantum collision attacks against double block length hash functions. We apply a quantum version of rebound attack that finds free-start collisions on Hirose's DBL compression function instantiated with AES-256 (in short, we call it HCF-AES-256). The proposed attack covers up to 10-rounds of HCF-AES-256 in the quantum setting. Our rebound attack uses two inbound phases that help us to mount dedicated quantum collision attacks against 10-round HCF-AES-256 that are faster than the generic quantum collision attacks even when small qRAM or no qRAM is available.

Table 1: Classical and quantum free-start collision attacks on HCF-AES-256. Q-Model-I and Q-Model-II denote quantum-attack settings with qRAM and without qRAM, respectively. P denotes the number of classical processors. S denotes the size of quantum computer in qubits. TSTO denotes the time-space trade-off setting.

Settings	Attack	Rounds	Time	Space	qRAM	Reference
Classical	Dedicated	8	2^{96}	2^8	0	[CHKM16]
Classical	Dedicated	9	2^{120}	2^8	0	[CHKM16]
Classical	Generic	all	2^{128}	0	0	Birthday paradox
Classical (parallel-rho)	Generic	all	$\frac{2^{128}}{P}$	P	0	§ 2.4
Quantum (Q-Model-I)	Dedicated	10	$2^{85.11}$	0	2^{16}	§ 3
Quantum (Q-Model-I)	Generic	all	2^{120}	0	2^{16}	[BHT98]
Quantum (Q-Model-I)	Generic	all	$2^{85.33}$	0	$2^{85.33}$	[BHT98]
Quantum (Q-Model-II)	Dedicated	10	$2^{88.61}$	0	0	§ 4.1
Quantum (Q-Model-II)	Dedicated	10	$2^{86.07}$	0	0	§ 4.2
Quantum (Q-Model-II)	Dedicated (TSTO)	10	$\frac{2^{88.61}}{\sqrt{S/2^4}}$	$2^4 \leq S < 2^{76}$	0	§ 4.3
Quantum (Q-Model-II)	Generic	all	$2^{102.4}$	$2^{51.2}$	0	[CNS17]

However, the success of our attack largely depends on the configuration of the 16-byte constant (used in the design of HCF-AES-256, cf. § 2.2): the number of its non-zero bytes and their positions. On the other hand, the best publicly known classical attack exists up to 9-rounds of HCF-AES-256 if 4-bytes of the constant c are non-zero. Moreover, our attack improves the flexibility of the previously known best attack by allowing 8-bytes of constant c to be non-zero at some specific positions.

In addition, we propose a MILP-based method to systematically explore the search space of useful differential trails for the rebound attack with multiple inbound phases. Using this method, we find a differential trail for 10-round AES-256 with differential probability 2^{-160}. We demonstrate that this trail can be used to mount free-start collision attacks on 10-round HCF-AES-256 in the quantum setting when a small qRAM is available and the 16-byte constant has eight non-zero bytes at some specific positions. We also present quantum collision attacks on 10-round HCF-AES-256 when no qRAM is available.

In time-space trade-off (TSTO) setting, if a quantum computer of size S qubits is available, then we can find collisions with time complexity $2^{88.61}/\sqrt{S/2^4}$, where $2^4 \leq S < 2^{76}$. A summary of free-start collision attacks against HCF-AES-256 is given in Table 1.

1.2 Organization of the Paper

In § 2, we give preliminaries on Hirose's DBL compression function, AES, quantum computations, generic quantum collision attack settings and rebound attack. In § 3, following the rebound attack procedure, we present a quantum free-start collision attack on 10-round HCF-AES-256 when the qRAM available is small. In § 4, we show that the attack described earlier can be slightly modified, yet still remaining valid, in the setting when we do not have any qRAM. In § 4.4, we briefly discuss the quantum free-start collision attack on HCF-AES-192. In § 5, we describe how to search for useful truncated differential trails by MILP methods with multiple inbound phases. Finally, in § 6, we conclude and outline some directions for future work in this line of work.

2 Preliminaries

This section gives a brief introduction of AES-256, Hirose's double block length compression function, basic quantum computation and quantum random access memories (qRAMs), the frameworks for generic quantum collision-finding attacks, and the quantum version of rebound attacks.

2.1 Description of AES-256

AES-256 is a NIST/ISO standardized iterated block cipher which encrypts 128-bit plaintexts with 256-bit keys. The 128-bit block is arranged into a 4×4 byte matrix, whose bytes are numbered as described in Figure 1. AES-256 has 14 rounds, where each round function, except the last, consists of four subroutines in the following order:

- SubBytes (SB) is a non-linear byte-wise substitution that applies the same 8×8 Sbox S to every byte.
- ShiftRows (SR) is a cyclic shift of the i^{th} row by i bytes to the left.
- MixColumns (MC) is a matrix multiplication over a finite field applied to each column.
- AddRoundKey (ARK) is an exclusive-or with the round subkey.

Before the first round, an additional whitening ARK operation is performed, and the last round executes without MC operation.

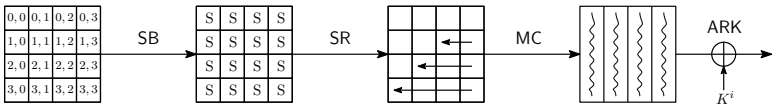

Figure 1: The round function of AES

Next, we describe the key schedule of AES-256. The 256-bit master key is divided into 8 words of 32 bits each $(W[0], W[1], \ldots, W[7])$. To generate the 15 subkeys of 128 bits (which consists of 60 words of 32 bits), the following key-expansion algorithm is used:

- For $i = 0, 1, \ldots, 59$ do the following:
  - If $i \equiv 0 \mod 8$, then $W[i] = W[i-8] \oplus \mathsf{SB}(\mathsf{RotByte}(W[i-1])) \oplus \mathsf{Rcon}[i/8]$,
  - If $i \equiv 4 \mod 8$, then $W[i] = W[i-8] \oplus \mathsf{SB}(W[i-1])$,
  - Else $W[i] = W[i-8] \oplus W[i-1]$,

where RotByte represents 1 byte rotation (i.e., $(a_0, a_1, a_2, a_3) \rightarrow (a_1, a_2, a_3, a_0)$), and Rcon denotes an array of fixed constants.

2.2 Hirose's Double Block Length Compression Function

Hirose's DBL compression function [Hir04, Hir06] internally evaluates a $2n$-bit keyed block cipher $E : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ by calling it two times. The first cipher call already compresses the entire input to the compression function, but the second cipher call also compresses the input independently of the first cipher call to produce a $2n$-bit output. Formally, Hirose's compression function can be defined as follows.

Definition 1. Let $F : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ be a compression function such that $(v_0, v_1) = F(h_0, h_1, M)$, where $h_0, h_1, M \in \{0,1\}^n$. The function F consists of a $(2n, n)$-bit block cipher $E : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ that produces the outputs as follows:

$$F(h_0, h_1, M) = (f_0(h_0, h_1, M), f_1(h_0, h_1, M)) = (v_0, v_1) \tag{1}$$

and the functions f_0 and f_1 are defined by

$$f_0(h_0, h_1, M) = E_{h_1||M}(h_0) \oplus h_0$$
$$f_1(h_0, h_1, M) = E_{h_1||M}(h_0 \oplus c) \oplus h_0 \oplus c,$$

where $||$ represents the concatenation and $c \in \{0,1\}^n \backslash \{0\}^n$ is a non-zero constant.

The DBL compression function given in Definition 1 is also shown in Figure 2.

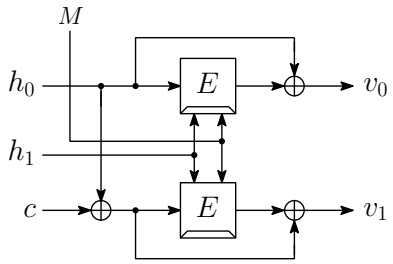

Figure 2: Hirose's double block length compression function CF.

2.3 Quantum Computation and Quantum RAM

A quantum computer applies quantum gates on inputs available in qubits to obtain new quantum states. A qubit ($|0\rangle$ or $|1\rangle$) is a quantum system defined over a finite set $B = \{0,1\}$. The state of a 2-qubit quantum system $|\psi\rangle$ is the superposition defined as $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. In general, the states of an n-qubit quantum system can be described as unit vectors in $\mathbb{C}^{2^n}$ under the orthonormal basis $\{|0\ldots00\rangle, |0\ldots01\rangle, \ldots |1\ldots11\rangle\}$, alternatively written as $\{|i\rangle : 0 \le i < 2^n\}$. Any quantum algorithm is described by a sequence of gates in the form of a quantum circuit, and all quantum computations are reversible. We use the standard quantum circuit model and adopt the Clifford group $\{H, CNOT, S = T^2\}$ plus T gates. Here, H is the single-qubit Hadamard gate $H : |b\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b\,|1\rangle)$, CNOT is the two-qubit controlled-NOT gate $CNOT : |a\rangle\,|b\rangle \mapsto |a\rangle\,|b \oplus a\rangle$), and T is the $\pi/8$ gate defined as $T : |0\rangle \mapsto |0\rangle$ and $T : |1\rangle \mapsto e^{i\pi/4}\,|1\rangle$. The identity operator on n-qubit states is denoted by I_n.

When we estimate time complexity of an attack on a primitive, we assume unit of time to be the time required to run the primitive once (e.g., the time required for one encryption if the primitive is a block cipher). In addition, when we estimate space complexity of a quantum attack on a primitive, we regard the number of qubits to implement the target primitive as the unit of space size.

Superposition Oracles for Classical Circuit. Considering a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the quantum oracle for f is the unitary transformation $\mathcal{U}_f$ acting on the $(n+1)$-qubit system that transforms a standard basis vector $|x\rangle\,|y\rangle \mapsto |x\rangle\,|y \oplus f(x)\rangle$, where $x \in \{0,1\}^n$ and $y \in \{0,1\}$. The linear operator $\mathcal{U}_f$ acts on the superposition states as

$$\mathcal{U}_f \left(\sum_{x\in\{0,1\}^n} a_i\,|x\rangle\,|0\rangle \right) = \sum_{x\in\{0,1\}^n} a_i\,|x\rangle\,|f(x)\rangle\,. \tag{2}$$

Note that $\mathcal{U}_f$ can be implemented efficiently in the standard quantum circuit model as long as there is an efficient reversible classical circuit that computes f. To build the quantum circuit for unitary operator $\mathcal{U}_f$, we first construct an efficient reversible circuit for f and substitute quantum gates for each of the reversible gates involved.

Grover's Search Algorithm. Given a search space of 2^n elements, say $\{x : x \in \{0,1\}^n\}$ and a Boolean function or predicate $f : \{0,1\}^n \to \{0,1\}$, the best classical algorithm with a black-box access to f requires about 2^n evaluations of the black-box oracle to identify x such that $f(x) = 1$ with probability one. In the quantum setting, Grover's search algorithm [Gro96] solves this problem with about $O(\sqrt{2^n})$ calls to a quantum oracle $\mathcal{U}_f$ that outputs $\sum_x a_x\,|x\rangle\,|y \oplus f(x)\rangle$ upon input of $\sum_x a_x\,|x\rangle\,|y\rangle$. Firstly, we construct a uniform superposition of states

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle\,,$$

by applying the Hadamard transformation $H^{\otimes n}$ to $|0\rangle^{\otimes n}$. We then iteratively apply the Grover operator $(2\,|\psi\rangle\,\langle\psi| - I)\mathcal{U}_f$ to $|\psi\rangle$ such that the amplitudes of those values x with $f(x) = 1$ are amplified. We then measure the resulting state which gives x such that $f(x) = 1$ with an overwhelming probability.

The exact complexity of the Grover search can be estimated by implementing the oracle circuit efficiently. It is thus essential to have a precise estimate of the quantum resources needed to implement the oracle. For example, the oracle circuit might require a large or small qRAM for its implementation.

Quantum Random Access Memories (qRAMs). The quantum random access memory (qRAM) is the quantum analogue of the classical random access memory (RAM), which uses n qubits to address a quantum superposition of 2^n memory cells. Given a list of classical data $L = \{x_0, \ldots, x_{2^n-1}\}$ with $x_i \in \{0,1\}^m$, the qRAM for L is modeled as an unitary operator $\mathcal{U}^L_{\mathsf{qRAM}}$ defined by

$$\mathcal{U}^L_{\mathsf{qRAM}} : |i\rangle_{\mathsf{Address}} \otimes |y\rangle_{\mathsf{Output}} \mapsto |i\rangle_{\mathsf{Address}} \otimes |y \oplus x_i\rangle_{\mathsf{Output}}\,, \tag{3}$$

where $i \in \{0,1\}^n$, $y \in \{0,1\}^m$, and $|\cdot\rangle_{\mathsf{Address}}$ and $|\cdot\rangle_{\mathsf{Output}}$ may be regarded as the address and output registers respectively. Therefore, we can access any quantum superposition of the data cells by using the corresponding superposition of address:

$$\mathcal{U}^L_{\mathsf{qRAM}} \left(\sum_i a_i\,|i\rangle \otimes |y\rangle \right) = \sum_i a_i\,|i\rangle\,|y \oplus x_i\rangle\,. \tag{4}$$

When we say that qRAM is available, we assume that a quantum gate that realizes the unitary operation (3) (for a list of classical data) is available in addition to basic quantum gates.

2.4 Frameworks for Quantum Collision-Finding Attacks

This section reviews the various frameworks depending upon the generic quantum collision-finding algorithms in the quantum settings. Suppose that we have a differential trail with probability p. In the classical setting, we can mount a collision attack requiring at least $1/p$ operations, and such a collision attack is faster than a generic attack (birthday paradox) only if $1/p < 2^{n/2}$ or $p > 2^{-n/2}$ holds. In the quantum setting, similar to the work [HS20], we consider the following scenarios:

1. **BHT Algorithm (the setting with qRAM).** Brassard, Høyer, and Tapp [BHT98] developed the generic quantum collision-finding algorithm. It finds collisions in time $O(2^{n/3})$ by making $O(2^{n/3})$ quantum queries when exponentially large qRAM is available. Let $f : \{0,1\}^n \to \{0,1\}^n$ be a random function. BHT consists of two steps. The first step performs a classical precomputation that chooses a subset $X \subset \mathbb{F}_2^n$ of size $|X| = 2^{n/3}$ and computes the value $f(x)$ for all $x \in X$ (which requires $O(2^{n/3})$ queries and $O(2^{n/3})$ time). The $2^{n/3}$ pairs $L = \{x, f(x)\}_{x \in X}$ are stored into qRAM so that they can be accessed in quantum superpositions. Then the second step performs Grover search to find $x' \in \{0,1\}^n \backslash X$ such that $(x, f(x)) \in L$ and $f(x) = f(x')$ for some $x \in X$, which runs in time $O(\sqrt{2^n/|L|}) = O(2^{n/3})$ on average. If we find such an $x' \in \{0,1\}^n \backslash X$ (and $x \in X$), it implies that we find a collision for f since $f(x') = f(x)$.

   Hence, if we have a differential trail with probability p, then we can mount a collision attack in time $\approx \sqrt{1/p}$. Such an attack is faster than the generic attack (BHT algorithm) if $\sqrt{1/p} < 2^{n/3}$. In other words, the attack is better than generic if $p > 2^{-2n/3}$ and a large qRAM is available.

2. **Tradeoffs between Time and Space.** From the viewpoint of time-space complexity, BHT [BHT98] is worse than the classical parallel rho method by Oorschot and Wiener [vOW94]. Roughly speaking, when P classical processors are available, the parallel rho method finds a collision in time $O(2^{n/2}/P)$. If a quantum computer of size $2^{n/3}$ without qRAM is available then we can run the parallel rho method on such a quantum computer and find a collision in time $2^{n/6}$. This is faster than using BHT. Let S denote the size of computational resources required for a quantum algorithm (i.e., S is the maximum size of quantum computers and classical memory), and T denote its time complexity. The tradeoff $T \cdot S = 2^{n/2}$ given by the parallel rho method is the best one even in the quantum setting.

   Thus, if we have a differential trail with probability p, then we can mount a collision attack using the rebound technique in time $T \approx T_{in} \cdot \sqrt{1/p}$, where T_{in} is the time to perform the inbound phase of size S_0. Such an attack is faster than generic attack (parallel rho method) if $p > T_{in}^2 S_0 2^{-n}$ holds. In addition, if a quantum computer of size $S \geq S_0$ is available, by parallelizing the Grover search for outbound phase, we obtain the tradeoff $T = T_{in} \cdot \sqrt{1/p}\sqrt{S_0/S}$, which is better than generic tradeoff $T = 2^{n/2}/S$ as long as $S < 2^n . p/(T_{in}^2 . S_0)$.

3. **Small Quantum Computer with Large Classical Memory.** Suppose that only a small quantum computer of polynomial-size in terms of the number of qubits required for designing the circuit is available, but we can use an exponentially large classical memory. In this scenario, Chailloux et al. [CNS17] showed that we could find a collision in time $O(2^{2n/5})$ with a quantum computer of size $O(1)$ and $O(2^{n/5})$ classical memory. The product of T and S becomes around $2^{3n/5}$, which is larger than $2^{n/2}$, but it is quite usual to consider a classical memory of size $O(2^{n/5})$, which is usually available. The algorithm by Chailloux et al. [CNS17] shows that we can obtain another better tradeoff between time and space if we treat the sizes of quantum hardware and classical hardware separately.

Therefore, if we have a differential trail with probability p, then we can mount a collision attack in time $\approx \sqrt{1/p}$. Such an attack is faster than the generic attack (BHT algorithm) if $\sqrt{1/p} < 2^{2n/5}$ (or $p > 2^{-4n/5}$) even when no qRAM is available.

2.5 Rebound Attacks with Quantum Computers

The rebound attack consists of two phases, called inbound and outbound phases. The inbound phase is an efficient meet-in-the-middle phase, which exploits the truncated differences and the available degrees of freedom in the internal state to fulfill the low probability parts in the middle of a differential characteristic. In the probabilistic outbound phase, the matches of the inbound phase are computed backward and forward to obtain an attack on the hash or compression function. Usually, the inbound phase is repeated many times to generate enough starting points (data pairs) respecting the inbound differential, which then propagates to the outbound differential to satisfy the full truncated differential trail.

To mount a rebound attack in quantum setting, we use Grover's algorithm to a search space by defining a Boolean function F which marks the elements of interest. Let $(\Delta_{in}, \Delta_{out})$ be the input-output difference with regard to the inbound differential. Let $F : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \rightarrow \{0,1\}$ be a Boolean function such that $F(\Delta_{in}, \Delta_{out}) = 1$ if and only if the following conditions holds:

1. (Inbound phase.) Given an input-output difference $(\Delta_{in}, \Delta_{out})$, obtain the corresponding starting points (pairs of messages (M, M') and $(\widetilde{M}, \widetilde{M'})$ that satisfy the differential trail $\Delta_{in} \rightarrow \Delta_{out}$).
2. (Outbound phase.) Propagate (M, M') and $(\widetilde{M}, \widetilde{M'})$ to the beginning and end of the cipher to check whether the differential transformations are satisfied; and compute the value of $F(\Delta_{in}, \Delta_{out})$.
3. Uncompute steps 1 and 2.

Extended Inbound Phases. The idea is to use all available degrees of freedom (from both states and subkeys) to extend the rebound attack for more rounds. Extended inbound phases consist of more than one independent inbound phase, which can be connected by choosing subkeys accordingly. In the outbound phase, we further extend the differential trail backward and forward by propagating the matching differences in the inbound phases to get a truncated differential path in each direction. The rebound attack's quantum setting remains the same as described above, except in the case of connecting inbound phases through the subkeys generated from the master key.

3 Quantum Collision Attacks on 10-Round HCF-AES-256 with Small qRAM

This section gives the proposed free-start collision attack against HCF-AES-256. To find a collision for the compression function CF, the attack uses the following fact from [CHKM14, CHKM16]:

Fact 1. Suppose that a collision for f_0 is caused by the pair $(h_0, (h_1, M))$ and $(h_0 \oplus \Delta h_0, (h_1, M))$, that is, $f_0(h_0, (h_1, M)) = f_1(h_0 \oplus \Delta h_0, (h_1, M))$ and assume that $\Delta h_0 = c$. Then a collision for f_1 is also caused by the same pair $(h_0, (h_1, M))$ and $(h_0 \oplus \Delta h_0, (h_1, M))$.

Using the above fact, the goal of finding collisions on CF reduces to finding collisions on f_0, for which we can proceed as follows:

1. Find a colliding pair of inputs $(h_0, (h_1, M))$ and $(h_0 \oplus \Delta h_0, (h_1, M))$.
2. Output the pair $(h_0, (h_1, M))$ and $(h_0 \oplus \Delta h_0, (h_1, M))$ if $\Delta h_0 = c$. Otherwise return to Step 1 and repeat the process.

Chen et al. [CHKM14, CHKM16] instantiate CF with AES-256, and find collisions for f_0 using the rebound attack procedure. The attack technique returns a pair of colliding inputs (h_0, h_1, M) and (h'_0, h_1, M) with difference $\Delta h_0 = h_0 \oplus h'_0 = c$ whose bytes are non-zero at the same position as the non-zero bytes of the constant c.

In this section, we present a new differential trail for 10-round AES-256 and demonstrate how to use the differential trail to mount rebound attacks on HCF-AES-256 in the small-qRAM quantum setting. Our attack finds a colliding pair of inputs if the constant c has eight non-zero bytes at some specific positions.

3.1 A New Differential Trail for 10-Round AES-256

Here, we give a new differential trail with the differential probability $p_{out} = 2^{-96}$ for 10-round AES-256 that can be used to find collisions against 10-round HCF-AES-256. With some effort, we can come up with a 10-round differential trail as shown in Figure 3. Here, each 4×4 square matrix shows the active byte pattern of the AES state. This trail gives $p_{out} = 2^{-96}$ since the probability of an 8-byte cancellation for the feed-forward operation is 2^{-64}. We then use this trail to mount rebound attacks on 10-round HCF-AES-256 in the quantum settings, that returns a pair of colliding inputs (h_0, h_1, M) and $(h_0 \oplus \Delta h_0, h_1, M)$. Further, we need the condition $\Delta h_0 = c$, where c has 8 non-zero bytes at some specific positions, and this can be achieved with probability 2^{-64}. Therefore, the overall time complexity of the attack is $2^{96} \times 2^{64} = 2^{160}$.

3.2 Differential Distribution Table of S-box

We precompute the differential distribution table (DDT) of the S-box in Table $\mathbb{T}$ using Algorithm 1, and load it into RAM. We can compute an input-output data pair through 1 DDT access on given an input-output difference to a cell. Since the S-box can be implemented with RAM, we regard that one random access to a classical memory or qRAM is equivalent to one S-box application.

Algorithm 1: The differential distribution table of S with data pairs

```
Let 𝕋 be an empty dictionary.
for δ_in ∈ F_2^8 do
    for x ∈ F_2^8 do
        x' ← x ⊕ δ_in, y ← S(x), y' ← S(x'), δ_out ← y ⊕ y'
        if x ≤ x' then
            Insert (x, x', y, y') into 𝕋[δ_in, δ_out]
return 𝕋
```

3.3 A Small-qRAM Collision Attack on 10-Round HCF-AES-256

At the core of the attack, we apply Grover's algorithm to a search space where an efficiently computable Boolean function marks the elements of interest to be possible solutions. Next, we proceed to define our Boolean function F.

We assume that the instantiated input-output difference pair is represented as $(\Delta_{in}, \Delta_{out})$ for the inbound differential with regard to Figure 3. The goal of the

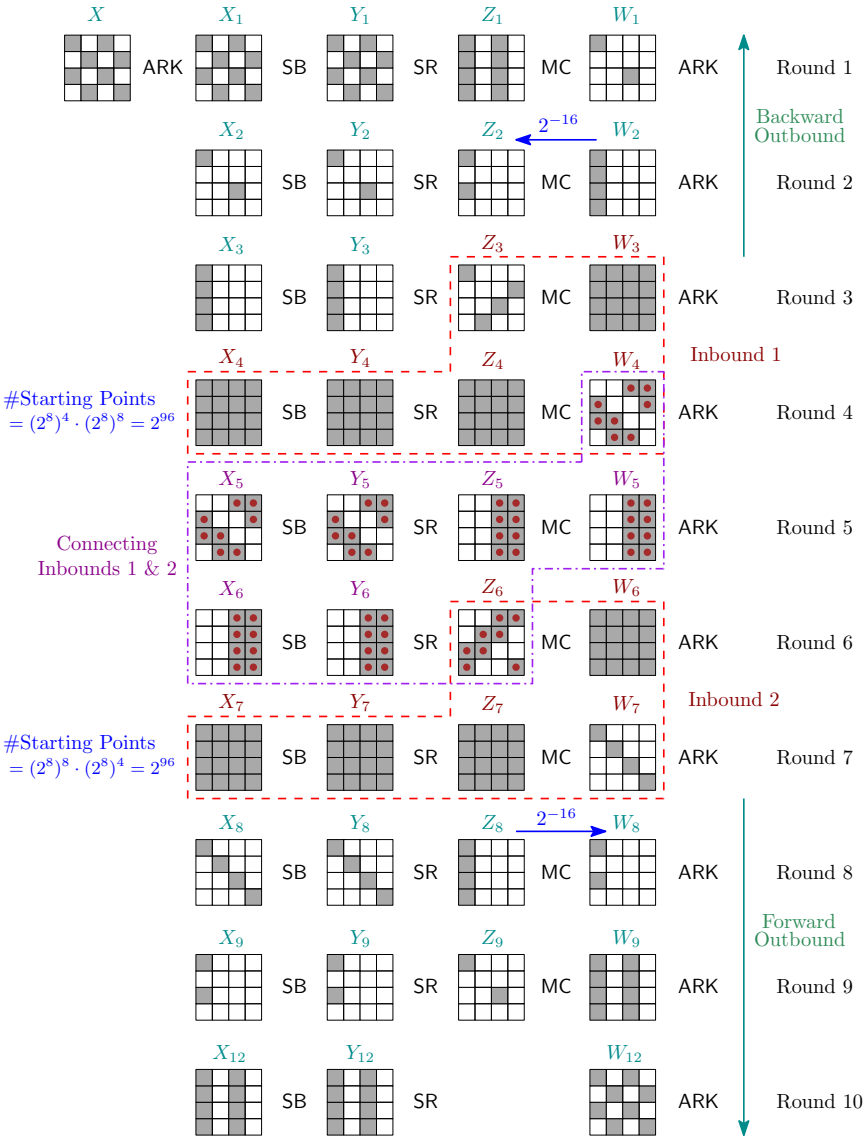

Figure 3: A Differential Trail for 10-Round AES-256

inbound phase of a rebound attack is to generate data pairs respecting the multiple inbound differentials. For two inbound phases, we define the input-output difference pair $(\Delta_{in}, \Delta_{out}) = (\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out})$, where $(\Delta^1_{in}, \Delta^1_{out})$ be the input-output difference pair for the first inbound differential, and $(\Delta^2_{in}, \Delta^2_{out})$ be the input-output difference pair for the second inbound differential. For the complete inbound differential, we define a Boolean function

$$F : \mathbb{F}_2^{32} \times \mathbb{F}_2^{48} \times \mathbb{F}_2^{48} \times \mathbb{F}_2^{32} \to \mathbb{F} \tag{5}$$

in a way such that $F(\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out}) = 1$ if and only if the starting point computed with $(\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out})$ fulfils the backward and forward outbound differentials. Therefore, if $F(\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out}) = 1$, we can produce two different colliding inputs h_0 and h'_0 such that

$$\mathsf{CF}(h_0, (h_1, M)) = \mathsf{CF}(h'_0, (h_1, M)),$$

where h_0 and h'_0 are obtained from the starting point, and (h_1, M) is obtained from the keys derived from connecting rounds in inbound phases 1 and 2.

On given $(\Delta_{in}, \Delta_{out}) = (\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out})$, the function $F(\Delta_{in}, \Delta_{out}) = F(\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out})$ can be computed with a classical computer by the following approach:

1. Compute the differential $(\Delta X_4^{(i)}, \Delta Y_4^{(i)})$ for each S-box $S_4^{(i)}$ $(0 \leq i < 16)$ from $(\Delta Z_3, \Delta W_4)$ in the fourth round, where $\Delta Z_3 = \Delta^1_{in}$ and $\Delta W_4 = \Delta^1_{out}$.

2. Solve the active S-box $S_4^{(i)}$ to obtain $X_4^{(i)}$ for $i \leq 0 < 16$, such that

$$S_4^{(i)}(X_4^{(i)}) \oplus S_4^{(i)}(X_4^{(i)} \oplus \Delta X_4^{(i)}) = \Delta Y_4^{(i)}.$$

   Pick $\min\{X_4^{(0)}, X_4^{(0)} \oplus \Delta X_4^{(0)}\}$ as the new value for $X_4^{(0)}$. Similarly, we obtain $X_4^{(1)}, X_4^{(2)}, \ldots, X_4^{(15)}$. We can build the starting point as

$$X_4 = (X_4^{(0)}, X_4^{(1)}, \ldots, X_4^{(15)}).$$

   If there are no admissible inputs for the pair $(\Delta X_4, \Delta Y_4)$, then return to Step 1.

3. Compute the differential $(\Delta X_7^{(i)}, \Delta Y_7^{(i)})$ for each S-box $S_7^{(i)}$ $(0 \leq i < 16)$ from $(\Delta Z_6, \Delta W_7)$ in the seventh round, where $\Delta Z_6 = \Delta^2_{in}$ and $\Delta W_7 = \Delta^2_{out}$.

4. Solve the active S-box $S_7^{(0)}$ to obtain $X_7^{(0)}$ such that

$$S_7^{(0)}(X_7^{(0)}) \oplus S_7^{(0)}(X_7^{(0)} \oplus \Delta X_7^{(0)}) = \Delta Y_7^{(0)}.$$

   Pick $\min\{X_7^{(0)}, X_7^{(0)} \oplus \Delta X_7^{(0)}\}$ as the new value for $X_7^{(0)}$. Similarly, we obtain $X_7^{(1)}, X_7^{(2)}, \ldots, X_7^{(15)}$. We can build the starting point as

$$X_7 = (X_7^{(0)}, X_7^{(1)}, \ldots, X_7^{(15)}).$$

   If there are no admissible inputs for the pair $(\Delta X_7, \Delta Y_7)$, then return to Step 3.

5. Select ΔY_5 compatible with ΔX_5 with the help of precomputed DDT lookup table. Calculate ΔX_6 form ΔY_5 and check whether ΔX_6 and ΔY_6 are compatible for each eight active bytes. If there are no admissible inputs for the pair $(\Delta X_5, \Delta Y_5)$ and $(\Delta X_6, \Delta Y_6)$, then repeat the process in Step 5.

6. Connect the results of two inbound phases to ensure that the differences in the eight active bytes of round 5 and the actual values of Y_5 and X_6 match by choosing the subkeys K_4, K_5 and K_6 accordingly.

7. Using the key schedule of AES-256, we compute the round key K_3 from K_4 and K_5, K_2 from K_3 and K_4, K_1 from K_2 and K_3, and K_0 from K_1 and K_2. Similarly, we can compute the round key K_7 from K_5 and K_6, K_8 from K_6 and K_7, K_9 from K_7 and K_8, and K_{10} from K_8 and K_9.

8. Compute starting points X_5 and X_6 from key K_4 with state W_4 and from key K_6 with state W_5 respectively. Note that we now have a corrected path for the starting points $X_4 \to X_5 \to X_6 \to X_7$.

9. If the starting point $(X_4, X_4 \oplus \Delta X_4)$ obtained in Step 2 respects the backward outbound differential, and the starting point $(X_7, X_7 \oplus \Delta X_7)$ obtained in Step 4 respects the backward outbound differential, then $F(\Delta_{in}, \Delta_{out})$ returns 1; otherwise it returns 0.

At the end, if $F(\Delta_{in}, \Delta_{out}) = 1$, we compute the corresponding inputs $(X, X \oplus \Delta X)$ from X_4 and ΔX_4, and $(W_{12}, \Delta W_{12})$ from Y_7 and ΔY_7. If $\Delta X = \Delta W_{12}$, then output the pair of inputs (K, X) and $(K, X \oplus \Delta X)$, which are mapped to the same hash value by f_0 instantiated with 10-round AES-256, where $K = K_0||K_1$.

Therefore, by applying Grover's search with the quantum oracle $\mathcal{U}_F$ which maps $|\Delta_{in}, \Delta_{out}, \alpha\rangle |y\rangle$ to $|\Delta_{in}, \Delta_{out}, \alpha\rangle |y \oplus F(\Delta_{in}, \Delta_{out}, \alpha)\rangle$, we can find a collision with around $\frac{\pi}{4}.\sqrt{2^{160}}$ queries. To estimate the overall complexity, we need to find the exact complexity incurred by $\mathcal{U}_F$.

3.4 Implementation of the Quantum Oracle $\mathcal{U}_F$

To implement the quantum oracle $\mathcal{U}_F$, we firstly define an additional function $D^{(i)}$ for $0 \leq i < 16$, that computes the actual input-output data pair respecting the differential of each S-box S for round j by accessing the precomputed DDT. For example, the function $D^{(i)}$ outputs $\min\{X_4^{(i)}, X_4^{(i)} \oplus \Delta X_4^{(i)}\}$ upon input $(\Delta X_4^{(i)}, \Delta Y_4^{(i)})$ such that $S(X_4^{(i)}) \oplus S(X_4^{(i)} \oplus \Delta X_4^{(i)}) = \Delta Y_4^{(i)}$.

Since the computation of $D^{(i)}$ in the classical setting uses the table $\mathbb{T}$ computed by Algorithm 1, implementing a quantum oracle of $D^{(i)}$ requires qRAMs of 2^{16} size. Thus, the oracle $\mathcal{U}_F$ can be constructed with the quantum circuit of $D^{(i)}$, which is presented in Algorithm 2.

3.5 Computing Round Key K_6

The following eight conditions are deduced from the AES-256 key expansion algorithm:

$$K_6[0][0] = K_4[1][0] \oplus S(K_5[1][3]) \tag{6}$$
$$K_6[1][0] = K_4[2][0] \oplus S(K_5[2][3]) \tag{7}$$
$$K_6[0][1] \oplus K_6[0][2] = K_4[0][2] \tag{8}$$
$$K_6[0][2] \oplus K_6[0][3] = K_4[0][3] \tag{9}$$
$$K_6[1][2] \oplus K_6[1][3] = K_4[1][3] \tag{10}$$
$$K_6[2][0] \oplus K_6[2][1] = K_4[2][1] \tag{11}$$
$$K_6[3][0] \oplus K_6[3][1] = K_4[3][1] \tag{12}$$
$$K_6[3][1] \oplus K_6[3][2] = K_4[3][2] \tag{13}$$

Notice that all the bytes of K_4 and K_5 on the right side are already fixed by the algorithm. The other conditions come from the fixed bytes of $Y_6[2]$ and $Y_6[3]$:

$$\mathsf{SR}(Y_6[2]) = \mathsf{MC}^{-1}(X_7)[2] \oplus \mathsf{MC}^{-1}(K_6)[2] \tag{14}$$
$$\mathsf{SR}(Y_6[3]) = \mathsf{MC}^{-1}(X_7)[3] \oplus \mathsf{MC}^{-1}(K_6)[3] \tag{15}$$

Now since X_7 is given to the algorithm as input, the above equations (14) and (15) can be expanded as the following linear equations:

$$(0d, 09, 0e, 0b)K_6[0] = Z_6[2][0] \oplus (0d, 09, 0e, 0b)X_7[0] \tag{16}$$
$$(09, 0e, 0b, 0d)K_6[1] = Z_6[1][1] \oplus (09, 0e, 0b, 0d)X_7[1] \tag{17}$$
$$(0e, 0b, 0d, 09)K_6[2] = Z_6[0][2] \oplus (0e, 0b, 0d, 09)X_7[2] \tag{18}$$
$$(0b, 0d, 09, 0e)K_6[3] = Z_6[3][3] \oplus (0b, 0d, 09, 0e)X_7[3] \tag{19}$$
$$(0b, 0d, 09, 0e)K_6[0] = Z_6[3][0] \oplus (0b, 0d, 09, 0e)X_7[0] \tag{20}$$
$$(0d, 09, 0e, 0b)K_6[1] = Z_6[2][1] \oplus (0d, 09, 0e, 0b)X_7[1] \tag{21}$$
$$(09, 0e, 0b, 0d)K_6[2] = Z_6[1][2] \oplus (09, 0e, 0b, 0d)X_7[2] \tag{22}$$
$$(0e, 0b, 0d, 09)K_6[3] = Z_6[0][3] \oplus (0e, 0b, 0d, 09)X_7[3] \tag{23}$$

Algorithm 2: Implementation of $\mathcal{U}_F$

Input: $|\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out}\rangle\,|y\rangle$
Output: $|\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out}\rangle\,|y \oplus F(\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out})\rangle$

/* Perform inbound phase 1 */
for $i \in \{0, 1, \ldots, 15\}$ **do**
  Compute the corresponding differential $\Delta X_4^{(i)} \to \Delta Y_4^{(i)}$ for each S-Box from $(\Delta Z_3, \Delta W_4)$, where $\Delta Z_3 = \Delta^1_{in}$ and $\Delta W_4 = \Delta^1_{out}$.
  Run $D^{(i)}(\Delta X_4^{(i)}, \Delta Y_4^{(i)})$. Let $(X_4^{(i)}, X_4^{(i)} \oplus \Delta X_4^{(i)})$ be the output.
/* Perform inbound phase 2 */
for $i \in \{0, 1, \ldots, 15\}$ **do**
  Compute the corresponding differential $\Delta X_7^{(i)} \to \Delta Y_7^{(i)}$ for each S-box from $(\Delta Z_6, \Delta W_7)$, where $\Delta Z_6 = \Delta^2_{in}$ and $\Delta W_7 = \Delta^2_{out}$.
  Run $D^{(i)}(\Delta X_7^{(i)}, \Delta Y_7^{(i)})$. Let $(X_7^{(i)}, X_7^{(i)} \oplus \Delta X_7^{(i)})$ be the output.
Compute ΔX_5 from ΔW_4 and ΔY_6 from ΔZ_6.
/* Connect inbound phases 1 and 2 */
for $i \in \{1, 2, 6, 7, 8, 11, 12, 13\}$ **do**
  Run D^i for each $\Delta X_5^{(i)}$ to select the corresponding differential $\Delta Y_5^{(i)}$.
  Compute ΔX_6 from ΔY_5.
  for $j \in \{8, 9, 10, 11, 12, 13, 14, 15\}$ **do**
    Run $D^{(j)}(\Delta X_6^{(j)}, \Delta Y_6^{(j)})$.
    if ΔX_6 *and* ΔY_6 *are compatible* **then**
      return $(X_6^{(i)}, X_6^{(i)} \oplus \Delta X_6^{(i)})$ as the output.
  Let $(X_5^{(i)}, X_5^{(i)} \oplus \Delta X_5^{(i)})$ be the output.
Compute $X_5[0][2], X_5[0][3], X_5[1][0], X_5[1][3], X_5[2][0], X_5[2][1], X_5[3][1]$ and $X_5[3][2]$.
Compute $K_4[0][2], K_4[0][3], K_4[1][0], K_4[1][3], K_4[2][0], K_4[2][1], K_4[3][1]$ and $K_4[3][2]$ from the corresponding bytes of W_4 and X_5.
Compute $K_5[2]$ and $K_5[3]$ from the corresponding bytes of W_5 and X_6.
Compute the round key K_6 satisfying the conditions obtained so far by expanding them into 16 linear equations as described in § 3.5.
Compute the remaining bytes of K_4 from $K_5[2]$, $K_5[3]$ and K_6.
Compute W_5 from Y_4 with K_4, and X_6 from X_7 with K_6. Then compute $K_5[j] = W_5[j] \oplus X_6[j]$ for $0 \le j \le 1$.
Compute the round keys $K_0, K_1, K_2, K_3, K_7, K_8$, K_9 and K_{10}.
/* Create starting points derived from $(\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out})$ */
Set $X_4 \leftarrow (X_4^{(0)}, \ldots, X_4^{(15)})$ and $X'_4 \leftarrow (X_4^{(0)} \oplus \Delta X_4^{(0)}, \ldots, X_4^{(15)} \oplus X_4^{(15)})$
Set $X_7 \leftarrow (X_7^{(0)}, \ldots, X_7^{(15)})$ and $X'_7 \leftarrow (X_7^{(0)} \oplus \Delta X_7^{(0)}, \ldots, X_7^{(15)} \oplus X_7^{(15)})$
if (X_4, X'_4) *fulfils the backward outbound differential* **then**
  set 1-bit flag $\mathsf{flag}_1 = 1$; otherwise set $\mathsf{flag}_1 = 0$.
if (X_7, X'_7) *fulfils the forward outbound differential* **then**
  set 1-bit flag $\mathsf{flag}_2 = 1$; otherwise set $\mathsf{flag}_2 = 0$.
if $\mathsf{flag}_1 = 1$ *and* $\mathsf{flag}_2 = 1$ **then**
  return $|\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out}\rangle\,|y \oplus 1\rangle$
else
  return $|\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out}\rangle\,|y\rangle$

Observe that we can compute $K_6[0]$ from equations (6), (7), (16) and (20). Using equations (11) and (12), we can compute $K_6[2][1]$ and $K_6[3][1]$. Then we can compute $K_6[1]$ from equations (17) and (21). Similarly, we can compute $K_6[2]$ and $K_6[3]$, which allows us to compute the key K_6 fully.

3.6 Complexity Analysis

First, we describe the following facts and assumptions used in our quantum collision attack on HCF-AES-256.

- The complexity of the computation of 10-round AES is approximated by $(16+4)\times 10 = 200$ S-box computations.
- The complexity of one access to the qRAM that stores a table of input-output differences is equivalent to one S-box computation.
- The complexity of solving linear equations involved in computing K_6, which are conditioned on K_4 and K_5, is ignored.
- One computation of inverse S-box is about the same as computing two S-boxes [JNRV20].
- Uncomputing is taken into account to free-up the wires after executing a task.

In our attack setting, we first precompute the differential distribution table (DDT) with 2^{16} classical data for the S-box (see Algorithm 1) and then load this table into a qRAM in advance. This qRAM is accessed by the quantum circuit for $D^{(i)}$.

Complexity of $D^{(i)}$. $D^{(i)}$ is used to compute input-output data pairs through the precomputed DDT accesses as given in Algorithm 1. One DDT access is equivalent to one S-box evaluation. Hence, we need only one S-box evaluation, which is about $2\times\frac{1}{200} \approx 2^{-6.64}$ 10-round AES-256 computations.

Complexity of $\mathcal{U}_F$. In Algorithm 2, Steps 2-4 as well as Steps 6-8, make 16 calls to $D^{(i)}$. Steps 11-18 make $2^8 \times 16$ calls to $D^{(i)}$ since there might be 2^8 repetitions of Steps 10-25 for connecting two inbound phases. In Step 9, we compute backwards for the S-box in round 6, therefore, we need $2 \times 1 \times 16 = 32$ S-box computations. In Step 27, according to Figure 3, we need to compute backwards for 2 rounds and in Step 28, we need to compute forwards for 3 rounds from the starting points. Therefore, $2 \times 2 \times 16 = 64$ inverse S-box computations, and $2 \times 3 \times 16 = 96$ S-box computations are needed. Hence, we need $2 \times (32 \times 2 + 64 \times 2 + 96)/200 = 2.88$ 10-round AES-256 computations. The overall complexity of $\mathcal{U}_F$ is $16 \times 2^{-6.64} + 16 \times 2^{-6.64} + (2^8 \times 16 \times 2^{-6.64}) + 2.88 \approx 2^{5.46}$ 10-round AES-256 computations.

Complexity to find a collision. To identify a 160-bit value $(\Delta^1_{in}, \Delta^1_{out}, \Delta^2_{in}, \Delta^2_{out}) \in \mathbb{F}_2^{32} \times \mathbb{F}_2^{48} \times \mathbb{F}_2^{48} \times \mathbb{F}_2^{32}$ with Grover search such that $F(\Delta^1_{in}, \Delta^1_{out}, \Delta^2_{in}, \Delta^2_{out}) = 1$ requires $\frac{\pi}{4} \times \sqrt{2^{160}}$ queries to $\mathcal{U}_F$. Therefore, the complexity to find a collision is $\frac{\pi}{4} \times \sqrt{2^{160}} \times 2^{5.46} = 2^{85.11}$ 10-round AES-256 computations.

4 Quantum Collision Attacks on 10-Round HCF-AES-256 without qRAM

In this section, we introduce methods to eliminate the requirement of qRAM for implementing DDT. This permits us to use qubits only as working memory, without

requiring a dedicated qRAM.

We adopt three methods to get rid of the requirement of qRAM for accessing $D^{(i)}$. The first two methods are similar to the work by Dong et al. [DSS+20], while the third method relies on time-space tradeoff. Essentially, we re-implement $D^{(i)}$ without using the DDT stored in qRAMs, while keeping the functional behavior of $D^{(i)}$ unchanged.

4.1 Method 1: Using Grover's Search for S-box

The idea is to generate data pairs by online search instead of table lookups when given a specific input-output differential $(\delta_{in}, \delta_{out})$ for an 8×8 S-box. Specifically, we just replace the table lookups using $D^{(i)}$ by Grover's search algorithm for $D^{(i)}$ in Algorithm 2, while other parts of the algorithm remain the same.

Complexity of Grover Search for $D^{(i)}$. To apply Grover's algorithm to $D^{(i)}$ on given $(\Delta X_4^{(i)}, \Delta Y_4^{(i)})$, we define $D^{(i)}(\Delta X_4^{(i)}, \Delta Y_4^{(i)}, \cdot) : \mathbb{F}_2^8 \to \mathbb{F}_2$ to find $X_4^{(i)}$. It requires $\frac{\pi}{4} \times \sqrt{2^8} = 2^{3.58}$ queries to the oracle $\mathcal{U}_{D^{(i)}}$, which is equivalent to $\frac{\pi}{4} \times \sqrt{2^8} \times (1/200) = 2^{-4.05}$ 10-round AES-256 computations. Hence, the total complexity of the Grover search on $D^{(i)}$ is about $2 \times 2^{-4.05} = 2^{-3.05}$ 10-round AES-256 computations.

Complexity of $\mathcal{U}_F$. In Algorithm 2, the implementation of $\mathcal{U}_F$ without qRAM is obtained by replacing $D^{(i)}$ with their versions without qRAM. The overall complexity of $\mathcal{U}_F$ is $16 \times 2^{-3.05} + 16 \times 2^{-3.05} + (2^8 \times 16 \times 2^{-3.05}) + 2.88 \approx 2^{8.96}$ 10-round AES-256 computations.

Complexity to find a collision. To identify a 160-bit value $(\Delta_{in}^1, \Delta_{out}^1, \Delta_{in}^2, \Delta_{out}^2) \in \mathbb{F}_2^{32} \times \mathbb{F}_2^{48} \times \mathbb{F}_2^{48} \times \mathbb{F}_2^{32}$ with Grover search such that $F(\Delta_{in}^1, \Delta_{out}^1, \Delta_{in}^2, \Delta_{out}^2) = 1$ requires $\frac{\pi}{4} \times \sqrt{2^{160}}$ queries to $\mathcal{U}_F$. Therefore, the complexity to find a collision is $\frac{\pi}{4} \times \sqrt{2^{160}} \times 2^{8.96} = 2^{88.61}$ 10-round AES-256 computations.

4.2 Method 2: Using a Dedicated Quantum Circuit for S-box

At FSE 2020, Bonnetain et al. [BNS19b] proposed a quantum circuit that fulfils the functionality of DDT. The cost is equivalent to 2 S-box computations and 22 ancilla qubits. We adopt their idea to implement $\mathcal{U}_F$ without qRAMs. The complexity is quite similar to Algorithm 2, because when one DDT access is needed, we just replace it by 2 S-box evaluations. The updated complexity of $D^{(i)}$ is $2 \times (2/200) \approx 2^{-5.64}$ 10-round AES-256 computations. Hence the complexity of $\mathcal{U}_F$ becomes $16 \times 2^{-5.64} + 16 \times 2^{-5.64} + 2^8 \times 16 \times 2^{-5.64} + 2.88 \approx 2^{6.42}$ 10-round AES-256 computations. Overall, we need $\frac{\pi}{4} \times \sqrt{2^{160}} \times 2^{6.42} = 2^{86.07}$ 10-round AES-256 computations with 22 ancilla qubits.

4.3 Method 3: A Time-Space Tradeoff

Recall that the generic collision finding algorithm in this setting is the parallel-rho method, which gives the tradeoff $T.S = 2^{n/2}$, or equivalently $T = 2^{n/2}/S$. We regard the size (the number of qubits) required to implement the attack target (here, 10-round AES-256) as the unit of space size.

We again use the same differential trail with probability 2^{-160} for the outbound phase, and thus the domain size of F is 2^{160}. Following § 4.1, the cost of $\mathcal{U}_F$ is $\approx 2^{8.96}$ encryptions. In addition, we require some ancillary quantum registers to realize $\mathcal{U}_F$. In Algorithm 2, Steps 4, 8, 12 and 15 require some ancillary quantum registers to solve the S-box differential equation $S(x_i) \oplus S(x_i \oplus \delta_{in})$. The size of quantum register is $\approx 1/16$ units (as the block size of S-box is 1/16 of the internal state size of AES). To compute and store the values $(\delta_{in}^{(i)}, \delta_{out}^{(i)})$ for $1 \leq i \leq 16$ in Steps 3, 7, 12 and 15, we use ancillary quantum registers of

size $\approx 4 \times (1/16 \times (2 \times 1/16)) = 8$. We also have to use a quantum register to store x_i and another quantum register to compute $D^{(i)}$. Both of these are of sizes $\approx 1/16$ units. Thus, additional quantum registers of size $16 \times (1/16 + 1/16) = 2$ is required for Steps 2-18. In Steps 29 and 31, we use additional quantum registers of size ≈ 5. The 2 qubits required for flag_1 and flag_2 are quite small and we can ignore them. In total, we use additional quantum registers of size $(8 + 2 + 5) \approx 2^4$. Therefore, we can implement $\mathcal{U}_F$ on a quantum circuit in such a way that it runs in time around $2^{8.96}$ encryptions with 10-round AES, by using ancillary quantum register of size around 2^4.

When a quantum computer of size S ($S \geq 2^4$) is available and we use them to parallelize the Grover search, our rebound attack runs in time $T \approx 2^{88.61}/\sqrt{S/2^4} = 2^{90.61}/\sqrt{S}$. Therefore, our attack is better than the generic attack in the setting where the efficiency of a quantum algorithm is measured by the tradeoff between time T and space S as long as $2^4 \leq S < 2^{76}$, but it is worse than the generic attack in other settings.

4.4 Discussion on Quantum Collision Attack on HCF-AES-192

Previously, we presented Hirose's double block length compression function instantiated with AES-256, namely HCF-AES-256 in Figure 2. Similarly, we can define HCF-AES-192 by modifying the message length from 128-bit to 64-bit.

To mount quantum free-start collision attacks on HCF-AES-192, we can easily find a 10-round differential trail for AES-192 with the same probability as for AES-256 by using the MILP methods. More specifically, by modifying the differential trail for AES-256 given in Figure 3 for connecting the two inbound phases (from round 4 to round 6), we can find a valid differential trail for AES-192. We need to select the positions of active bytes after applying AddRoundKey operation such that we can efficiently recover the round keys K_4, K_5, K_6 using the key schedule of AES-192. As a result, the attack strategy against HCF-AES-192 would not only remain the same as described in § 3, but also the time complexity of quantum collision attacks remains the same, i.e. $2^{85.11}$ in the setting of Q-Model-I. Note that this method can attack 10 out of 12 rounds of HCF-AES-192.

5 Searching for Differential Trail with MILP Methods

We now describe a Mixed Integer Linear Programming (MILP) based tool proposed by Mouha et al. [MWGP11] to find the optimal differential trail for AES-256.

MILP Model. In order to find the optimal differential trail for our attack, we use a tool based on MILP. This model describes the propagation characteristics of the difference patterns using linear inequalities and defines an objective function to minimize the complexity of the collision attack. Specifically, for finding collisions for hash functions using the rebound attack technique, we need to modify the MILP model by Mouha et al. [MWGP11] by simply converting the collision search such that the active byte patterns of the first round input and the last round output are identical.

Assume that there is a differential trail for E_K of HCF-AES-256 with probability p whose input-output differences share a common value Δ. That is, let the differential trail be $E_{h_1||M}(h_0) \oplus E_{h_1||M}(h_0 \oplus \Delta) = \Delta$. Given around $1/p$ pairs of input messages with difference Δ, we expect one pair $((h_0, h_1, M), (h_0 \oplus \Delta, h_1, M))$ to follow this trail. Thus, the difference of the outputs of HCF-AES-256 for the valid pair of messages becomes zero, and leads to a collision.

Since K is known in hash functions, it is possible to generate many data pairs which conform to one particular segment of the desired trail. Then these pairs are tested to find the one which fulfills the remaining part of the trail. This is the basic strategy employed by the rebound attack [MRST09].

For each model, we fix the positions of two inbound phases. In the first inbound phase, we fix the round index $r+1$ for which MixColumns in rounds $r+1$ and $r+2$ are satisfied with cost one on average. Similarly, in the second inbound phase, we fix the round index $r+4$ in which MixColumns in rounds $r+4$ and $r+5$ are satisfied with cost one on average. We connect these two inbound phases in round $r+3$. Because the last round does not have MixColumns, we only have 4 choices in the case of the 10-round attack: $r \in \{1,2,3,4\}$ by starting the round counting from 0. For example, the 10-round trail introduced in § 3.1 is when $r=2$. The probability of the outbound phase is affected by two factors:

1. the number of difference cancellations in MixColumns,
2. the number of difference cancellations in the feed-forward.

Variables and Constraints. For an N-round primitive, we first introduce an integer variable r, which determines two inbound phases from round $r+1$ to $r+2$, and from round $r+4$ to $r+5$. These inbound phases are connected in round $r+3$. The backward outbound phase is connected from round r to 0, and the forward outbound phase is connected from round $r+5$ to $N-1$.

We introduce a set of 0-1 variables x_j for all cells of the states involved, where $x_j = 1$ if and only if the corresponding cell is differentially active. Let, $x_{i0}, x_{i1}, x_{i2}, x_{i3}$ denote the input bytes and $y_{i0}, y_{i1}, y_{i2}, y_{i3}$ denote the output bytes of the MixColumns transformation for each column. We also introduce a 0-1 dummy variable d to denote whether the column is active or not; and another variable b ($0 \le b \le 3$) to count the number of inactive bytes in active columns. Then, the proper relationships can be modeled in the following equality:

$$\begin{cases} -x_{i0} - x_{i1} - x_{i2} - x_{i3} + 4d = b & \text{for backward outbound phase} \\ -y_{i0} - y_{i1} - y_{i2} - y_{i3} + 4d = b & \text{for forward outbound phase} \end{cases}$$

Additionally, we use following set of the inequalities to model the behaviour of linear transformation of AES:

$$\begin{cases} x_{i0} + x_{i1} + x_{i2} + x_{i3} + y_{i0} + y_{i1} + y_{i2} + y_{i3} \ge \mathcal{B}_{\mathcal{D}} \cdot d \\ d - x_{i0} \ge 0 \\ d - x_{i1} \ge 0 \\ d - x_{i2} \ge 0 \\ d - x_{i3} \ge 0 \\ d - y_{i0} \ge 0 \\ d - y_{i1} \ge 0 \\ d - y_{i2} \ge 0 \\ d - y_{i3} \ge 0 \end{cases}$$

where $\mathcal{B}_{\mathcal{D}}$ denotes the differential branch number of the AES MixColumns transformation. The value of d is nonzero only if any of the $x_{i0}, x_{i1}, x_{i2}, x_{i3}, y_{i0}, y_{i1}, y_{i2}$ and y_{i3} is nonzero.

Finally, we need to add the constraints such that active byte patterns of the first round input and the last round output are identical. For example, to ensure the feed-forward cancellation for 10-round HCF-AES-256, we add the following constraints:

$$\begin{cases} X[0][0] = W_{12}[0][0]; X[0][1] = W_{12}[0][1]; X[0][2] = W_{12}[0][2]; X[0][3] = W_{12}[0][3]; \\ X[1][0] = W_{12}[1][0]; X[1][1] = W_{12}[1][1]; X[1][2] = W_{12}[1][2]; X[1][3] = W_{12}[1][3]; \\ X[2][0] = W_{12}[2][0]; X[2][1] = W_{12}[2][1]; X[2][2] = W_{12}[2][2]; X[2][3] = W_{12}[2][3]; \\ X[3][0] = W_{12}[3][0]; X[3][1] = W_{12}[3][1]; X[3][2] = W_{12}[3][2]; X[3][3] = W_{12}[3][3]; \end{cases}$$

where X and W_{12} (refer to Figure 3) denote the input and output differences of the 10-round differential trail for AES-256.

The Objective Function. To minimize the time complexity of the outbound phase, our objective function is to minimize the sum of b's and the variables x_0 to x_{15}.

Hence, our goal is to

$$\text{Minimize} \sum_{\text{Outbound}} b + \sum_{\text{Round 0}} x_j.$$

6 Conclusions and Open Problems

In this work, we presented quantum free-start collision attacks on the DBL compression function [Hir06] instantiated with 10-round AES-256, namely HCF-AES-256 when small qRAM or no qRAM is available. This is achieved by performing a quantum version of the rebound attack with extended inbound phases. Our attack on HCF-AES-256 outperforms the generic attack of Chailloux, Naya-Plasencia, and Schrottenloher [CNS17] in a model when large qRAM is not available. However, our attack has two limitations: (1) it is a free-start collision attack, and (2) we require the constant c to have a low hamming weight. More precisely, c should have 8 non-zero bytes at some specific positions for our attack to be valid. Extending these attacks to more than 10 rounds will be interesting.

Another interesting future work will be to extend the quantum collision attacks on Hirose's double block length compression function with other variants as given in [Hir06]. One might also revisit previous differential trail search activities so that we will be able to construct more efficient dedicated quantum collision-finding attacks against hash functions. The extension of the collision attack with real IV's remains an open problem.

Discussion on Related-Key Differential Cryptanalysis of Hash Functions based on AES. In a related-key attack against a block cipher, the attacker is given access to the encryption oracle under keys that differ from the target key by a known difference. In differential cryptanalysis, the attacker is allowed to introduce difference $\Delta X = X \oplus X'$ in plaintext pairs, whereas in related-key differential cryptanalysis, the attacker is additionally allowed to introduce difference $\Delta K = K \oplus K'$ in keys such that ΔX becomes ΔX_r after r rounds with high probability.

At present, in the classical setting, the best related-key differential attack [BKN09] can break the full 14-round AES-256 with total complexity of 2^{131} time and 2^{65} memory. Biryukov et al. [BKN09] also discussed how the related-key differential trail of AES-256 can be used to find free-start collisions for the Davies-Meyer compression function. The differences in the IV can be cancelled by the feed-forward operation of Davies-Meyer mode, if the difference in the plaintexts is equal to the difference in the ciphertexts.

Note that Hirose's double block length compression function uses the Davies-Meyer mode. This allows us to use a related-key differential for the underlying block cipher. The related-key differential trail for full AES-256 from [BKN09] does not have the same difference in the plaintext and the ciphertext pairs, which is required for cancellation by the feed-forward operation in this mode. Assuming that a related-key differential trail with same plaintext-ciphertext difference can be constructed for full AES-256 with probability 2^{-131} (same as that given in [BKN09]), we can find collisions on full 14-round HCF-AES-256. However, to mount the collision attack on HCF-AES-256, we need to satisfy an extra condition $\Delta h_0 = c$. If the attack requires c to have 8 non-zero bytes at some specific positions, as in the attacks described by us earlier, it will contribute a probability 2^{-64} cost to the attack. Therefore, the time complexity of quantum free-start collision attack will be $\approx \sqrt{2^{195}} = 2^{97.5}$. On the other hand, if we apply CNS algorithm [CNS17] on full HCF-AES-256, then we can find collisions with time complexity $2^{102.4}$ and $2^{51.2}$ classical memory. Quantifying the exact time/memory/data complexity of this attack, and improving it further to reduce the cost of the attack for full HCF-AES-256 in the related-key attack model remains an interesting research problem for future.

Acknowledgments

We would like to thank the anonymous reviewers of IACR Transactions on Symmetric Cryptology 2021 for their insightful comments and suggestions, which have significantly improved the presentation and technical quality of this work. We would also like to thank Dr. Kai-Min Chung for initial discussions on quantum computing which led to the initiation of this work. The third author would also like to thank MATRICS grant 2019/001514 by the Science and Engineering Research Board (SERB), Dept. of Science and Technology, Govt. of India for supporting the research carried out in this work.

References

[AFK+11] Frederik Armknecht, Ewan Fleischmann, Matthias Krause, Jooyoung Lee, Martijn Stam, and John P. Steinberger. The preimage security of double-block-length compression functions. In *ASIACRYPT*, volume 7073, pages 233–251. Springer, 2011.

[BHN+19] Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. Quantum attacks without superposition queries: The offline Simon's algorithm. In *ASIACRYPT*, volume 11921, pages 552–583. Springer, 2019.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN'98*, volume 1380, pages 163–169. Springer, 1998.

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In *CRYPTO*, volume 5677, pages 231–249. Springer, 2009.

[BNS19a] Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. On quantum slide attacks. In *Selected Areas in Cryptography - SAC*, volume 11959, pages 492–519. Springer, 2019.

[BNS19b] Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. Quantum security analysis of AES. *IACR Trans. Symmetric Cryptol.*, 2019(2):55–93, 2019.

[Bon17] Xavier Bonnetain. Quantum key-recovery on full AEZ. In *Selected Areas in Cryptography - SAC*, volume 10719, pages 394–406. Springer, 2017.

[BRSS10] John Black, Phillip Rogaway, Thomas Shrimpton, and Martijn Stam. An analysis of the block cipher-based hash functions from PGV. *J. Cryptol.*, 23(4):519–545, 2010.

[CHKM14] Jiageng Chen, Shoichi Hirose, Hidenori Kuwakado, and Atsuko Miyaji. A collision attack on a double-block-length compression function instantiated with round-reduced AES-256. In *Information Security and Cryptology - ICISC*, volume 8949, pages 271–285. Springer, 2014.

[CHKM16] Jiageng Chen, Shoichi Hirose, Hidenori Kuwakado, and Atsuko Miyaji. A collision attack on a double-block-length compression function instantiated with 8-/9-round AES-256. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 99-A(1):14–21, 2016.

[CNS17] André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *ASIACRYPT*, volume 10625, pages 211–240. Springer, 2017.

[DDW20] Xiaoyang Dong, Bingyou Dong, and Xiaoyun Wang. Quantum attacks on some Feistel block ciphers. *Des. Codes Cryptogr.*, 88(6):1179–1203, 2020.

[DSS+20] Xiaoyang Dong, Siwei Sun, Danping Shi, Fei Gao, Xiaoyun Wang, and Lei Hu. Quantum collision attacks on AES-like hashing with low quantum random access memories. In *ASIACRYPT*, volume 12492, pages 727–757. Springer, 2020.

[FGL09] Ewan Fleischmann, Michael Gorski, and Stefan Lucks. On the security of Tandem-DM. In *Fast Software Encryption - FSE*, volume 5665, pages 84–103. Springer, 2009.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *ACM Symposium on the Theory of Computing*, pages 212–219. ACM, 1996.

[Hir04] Shoichi Hirose. Provably secure double-block-length hash functions in a black-box model. In *Information Security and Cryptology - ICISC*, volume 3506, pages 330–342. Springer, 2004.

[Hir06] Shoichi Hirose. Some plausible constructions of double-block-length hash functions. In *Fast Software Encryption - FSE*, volume 4047, pages 210–225. Springer, 2006.

[HLMW93] Walter Hohl, Xuejia Lai, Thomas Meier, and Christian Waldvogel. Security of iterated hash functions based on block ciphers. In *CRYPTO*, volume 773, pages 379–390. Springer, 1993.

[HS18] Akinori Hosoyamada and Yu Sasaki. Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In *Topics in Cryptology - CT-RSA*, volume 10808, pages 198–218. Springer, 2018.

[HS20] Akinori Hosoyamada and Yu Sasaki. Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In *EUROCRYPT*, volume 12106, pages 249–279. Springer, 2020.

[IHM+19] Gembu Ito, Akinori Hosoyamada, Ryutaroh Matsumoto, Yu Sasaki, and Tetsu Iwata. Quantum chosen-ciphertext attacks against Feistel ciphers. In *Topics in Cryptology - CT-RSA*, volume 11405, pages 391–411. Springer, 2019.

[JNRV20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing grover oracles for quantum key search on AES and LowMC. In *EUROCRYPT*, volume 12106, pages 280–310. Springer, 2020.

[KLLN16] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In *CRYPTO*, volume 9815, pages 207–237. Springer, 2016.

[KM10] Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In *IEEE International Symposium on Information Theory*, pages 2682–2685. IEEE, 2010.

[LM92] Xuejia Lai and James L. Massey. Hash function based on block ciphers. In *EUROCRYPT*, volume 658, pages 55–70. Springer, 1992.

[LM17] Gregor Leander and Alexander May. Grover meets Simon - quantumly attacking the FX-construction. In *ASIACRYPT*, volume 10625, pages 161–178. Springer, 2017.

[LMS+15] Mario Lamberger, Florian Mendel, Martin Schläffer, Christian Rechberger, and Vincent Rijmen. The rebound attack and subspace distinguishers: Application to Whirlpool. *J. Cryptol.*, 28(2):257–296, 2015.

[Men17] Bart Mennink. Optimal collision security in double block length hashing with single length key. *Des. Codes Cryptogr.*, 83(2):357–406, 2017.

[Mer89] Ralph C. Merkle. One way hash functions and DES. In *CRYPTO*, volume 435, pages 428–446. Springer, 1989.

[MRST09] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *Fast Software Encryption - FSE*, volume 5665, pages 260–276. Springer, 2009.

[MS88] Carl H Meyer and Michael Schilling. Secure program load with manipulation detection code. In *Proc. Securicom*, volume 88, pages 111–130, 1988.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Information Security and Cryptology - Inscrypt*, volume 7537, pages 57–76. Springer, 2011.

[PGV93] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO*, volume 773, pages 368–378. Springer, 1993.

[Sho94] Peter W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In *Algorithmic Number Theory, ANTS-I*, volume 877, page 289. Springer, 1994.

[Sim97] Daniel R. Simon. On the power of quantum computation. *SIAM J. Comput.*, 26(5):1474–1483, 1997.

[Sta08] Martijn Stam. Beyond uniformity: Better security/efficiency trade-offs for compression functions. In *CRYPTO*, volume 5157, pages 397–412. Springer, 2008.

[Sta09] Martijn Stam. Blockcipher-based hashing revisited. In *Fast Software Encryption - FSE*, volume 5665, pages 67–83. Springer, 2009.

[vOW94] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with application to hash functions and discrete logarithms. In *CCS'94*, pages 210–218. ACM, 1994.