Quantum Rebound Attacks on Reduced-Round ARIA-Based Hash Functions

Seungjun Baek\textsuperscript{1} and Jongsung Kim\textsuperscript{1,2}

\textsuperscript{1} Department of Financial Information Security, Kookmin University, Republic of Korea
\{hellosj3, jskim\}@kookmin.ac.kr
\textsuperscript{2} Department of Information Security, Cryptology, and Mathematics, Kookmin University, Republic of Korea

\textbf{Abstract.} ARIA is a block cipher proposed by Kwon et al. at ICISC 2003, and it is widely used as the national standard block cipher in the Republic of Korea. In this study, we identify some flaws in the quantum rebound attack on 7-round ARIA-DM proposed by Dou et al., and we reveal that the limit of this attack is up to 5-round. Our revised attack applies not only to ARIA-DM but also to ARIA-MMO and ARIA-MP among the PGV models, and it is valid for all key lengths of ARIA. Moreover, we present dedicated quantum rebound attacks on 7-round ARIA-Hirose and ARIA-MJH for the first time. These attacks are only valid for the 256-bit key length of ARIA because they are constructed using the degrees of freedom in the key schedule. All our attacks are faster than the generic quantum attack in the cost metric of time–space tradeoff.

\textbf{Keywords:} Symmetric key cryptography · Block cipher-based hash function · Quantum attack · Rebound attack · ARIA

1 Introduction

ARIA \cite{22,21} is an iterative substitution permutation network (SPN) block cipher similar to AES \cite{7}, that supports a 128-bit block size and 128-, 192-, and 256-bit key lengths. Depending on the key lengths, it uses 12, 14, or 16 rounds. ARIA was presented by Kwon et al. at ICISC 2003 and was standardized by the Korean Agency for Technology and Standards. ARIA is described by RFC 5794 \cite{20} and has been supported by the Transport Layer Security protocol since 2011 \cite{30}. Since ARIA was developed, its security has been scrutinized by several cryptographers, and its full round security has not yet been broken, except by the use of the biclique attack, which is slightly faster than brute force.

Post-quantum cryptography received considerable attention after Shor’s seminal work \cite{32}, and NIST is in the process of selecting next-generation public-key schemes \cite{28}. Quantum computers have significantly influenced symmetric key schemes and hash functions, mainly by using Simon and Grover’s algorithms \cite{11,13}. In particular, Grover’s algorithm allows quantum computers to perform an exhaustive search on symmetric key schemes and hash functions.
with a quadratic speedup over the classical approach. Since 2015, the cryptography community has conducted extensive groundbreaking research, both theoretical and practical, including the analysis of block ciphers [2,19,14], hash functions [15,16,9,6], and permutations [12,26].

In a classical setting, the generic complexity required to find a collision of an \( n \)-bit hash function is \( O(2^{n/2}) \), according to the birthday paradox. In a quantum setting, the generic complexity of finding collisions depends on the settings for the resources available to the attacker. The BHT algorithm [4] finds collisions with a query complexity of \( O(2^{n/3}) \) when a quantum random access memory (qRAM) of \( O(2^{n/3}) \) is available. However, given the current state of development of quantum computers, it is highly probable that large qRAM will not be realized in the near future. A more realistic algorithm is the CNS algorithm proposed in 2017 by Chailloux et al. [5], which uses a large classical memory rather than a large qRAM. This CNS algorithm finds collisions with a classical memory of \( O(2^{n/5}) \), query complexity of \( O(2^{2n/5}) \), and quantum memory of only \( O(n) \). In both settings, the parallel rho method [29] gives the tradeoff time complexity \( T = 2^{n/2}/S \) when finding collisions.

**Related Works.** Hosoyamada and Sasaki [15] proposed a novel approach at Eurocrypt 2020. This approach showed that differential trails with a probability that are too low to be used for a rebound attack on hash functions in a classical setting are available in a quantum setting. They proposed quantum collision attacks on Matyas–Meyer–Oseas (MMO) and Miyaguchi–Preneel (MP) compression functions instantiated with AES that covered more rounds than those in a classical setting. Later, Dong et al. [9] improved the attacks on AES–MMO by significantly reducing the qRAM required for the attack. In ToSC 2021, quantum collision attacks on HCF–AES–256 and Simpira v2 were also proposed [6,26].

Dou et al. [10] proposed the first quantum rebound attack on the Davies–Meyer (DM) compression function when the underlying block cipher is instantiated with ARIA. Their attack was carried out by exploiting the degrees of freedom in states, and the probability of finding collisions was calculated considering the feed-forward operation. However, the complexity of finding collisions was inferior to the cost metric in any quantum setting currently considered, and the processes of constructing the attack using the degrees of freedom in the states were incorrect. Motivated by the work published in [15], we revised the above issues in detail and present attacks faster than the generic attacks in the cost metric of time–space tradeoff. We also took a closer look at algorithms for finding collisions with a quantum version of the rebound attack in several double block length (DBL) hash functions.

**Our Contributions.** In this study, we describe quantum rebound attacks on PGV, Hirose, and MJH instantiated with ARIA. For PGV models, DM, MMO, and MP constructions are primarily analyzed. Considering the structure of each compression function, our attack targets are divided into two categories: PGV (single block length (SBL) hash functions) and Hirose and MJH (DBL hash functions).
Quantum Rebound Attacks on Reduced-Round ARIA-Based Hash Functions

Table 1: Results of our attacks.

<table>
<thead>
<tr>
<th>Target</th>
<th>Construction Rounds</th>
<th>Type</th>
<th>Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIA-DM</td>
<td>SBL</td>
<td>Free-start collision</td>
<td>$2^{56.61}/\sqrt{S}$</td>
<td>Sect. 3</td>
</tr>
<tr>
<td>ARIA-MM0</td>
<td></td>
<td>Collision</td>
<td>$2^{119.83}/\sqrt{S}$</td>
<td>Sect. 4</td>
</tr>
<tr>
<td>ARIA-MP</td>
<td></td>
<td>Collision</td>
<td>$2^{119.67}/\sqrt{S}$</td>
<td></td>
</tr>
<tr>
<td>ARIA-Hirose</td>
<td>DBL</td>
<td>Free-start collision</td>
<td>$2^{119.83}/\sqrt{S}$</td>
<td></td>
</tr>
<tr>
<td>ARIA-MJH</td>
<td>DBL</td>
<td>Semi-free-start collision</td>
<td>$2^{119.67}/\sqrt{S}$</td>
<td></td>
</tr>
</tbody>
</table>

$S$ denotes the size of the quantum computer in qubits.

† In [10], an attack on reduced-round ARIA-DM was proposed. However, there are some flaws regarding the validity of the attack process and complexity, which we consider in Section 3.

We refer to the PGV hash functions as ARIA-DM, ARIA-MM0, and ARIA-MP, and to the Hirose/MJH hash functions as ARIA-Hirose, and ARIA-MJH.

We revised some issues of Dou et al.’s 7-round quantum rebound attack on ARIA-DM and found that, in fact, the attack is possible up to 5-rounds with some improved techniques. This attack can also be applied to ARIA-MM0 and ARIA-MP, with the same attack complexity as before. When $S$ quantum computers are available, the attack complexity is about $2^{56.61}/\sqrt{S}$. Since the generic attack complexity under the time–space metric is $2^{64}/S$, our attack is faster than the generic attack when $S < 2^{14.78}$.

We also extended the 5-round differential trail to 7-round. Our trail is constructed by exploiting $2^{128}$ degrees of freedom, which are only available in ARIA-256, and is mounted to find collisions of DBL hash functions such as Hirose and MJH. When $S$ quantum computers are available, the attack complexity is about $2^{119.83}/\sqrt{S}$ and $2^{119.67}/\sqrt{S}$ where $S < 2^{16.34}$ and $S < 2^{16.66}$ for Hirose and MJH, respectively.

Table 1 shows the details of the attack complexities on different targets.

Paper Organization. Section 2 describes ARIA, our attack target block cipher-based hash functions, and basic quantum computation. Section 3 briefly describes the rebound attack, the previous quantum rebound attack on 7-round ARIA-DM, and our revised quantum rebound attacks on 5-round ARIA-DM, ARIA-MM0, and ARIA-MP. Section 4 provides a new differential trail for 7-round ARIA and shows that it can be used to find collisions of DBL hash functions. Section 5 presents the conclusions.

2 Preliminaries

In this section, we briefly describe ARIA, our attack target block cipher-based hash functions, and the basic quantum computation required for our attacks.
2.1 ARIA

ARIA is a 128-bit block cipher with an SPN structure. The wide trail strategy of AES is used throughout its algorithm. ARIA can be used with three different key lengths: 128-, 192-, and 256-bit. Its number of rounds depends on the key length, with 12, 14, and 16 rounds for ARIA-128, ARIA-192, and ARIA-256, respectively. All states of the algorithm are treated as $4 \times 4$ matrices with elements in $GF(2^8)$ (Figure 1).

\[
\begin{array}{cccc}
  x_0 & x_1 & x_5 & x_9 \\
  x_1 & x_5 & x_9 & x_{13} \\
  x_2 & x_6 & x_{10} & x_{14} \\
  x_3 & x_7 & x_{11} & x_{15}
\end{array}
\]

Fig. 1: ARIA byte ordering

The round function of ARIA first applies a round key addition (RKA), followed by a substitution layer (SL), and then a diffusion layer (DL). An $R$-round ARIA repeats the round function $R - 1$ times, and in the last round, the diffusion layer is replaced with round key addition, which is the postwhitening key. The round function operations of ARIA are as follows:

**RKA.** The internal state is XORed with a 128-bit round key. The round keys are deduced from the master key via a key scheduling algorithm, which is described later in this section.

**SL.** A nonlinear 8-bit to 8-bit S-box is applied to each byte of the state. ARIA uses four S-boxes $S_1$ and $S_2$ and their inverses $S_1^{-1}$ and $S_2^{-1}$, respectively, where $S_1$ is the same as that of AES. In odd rounds, the S-boxes are applied, column-wise, in the order $(S_1, S_2, S_1^{-1}, S_2^{-1})$, whereas in even rounds, they are applied in the order $(S_1^{-1}, S_2^{-1}, S_1, S_2)$. Figure 2 describes the difference in SLs in odd and even rounds.

(a) SL in odd rounds

\[
\begin{array}{cccc}
  S_1 & S_1 & S_1 & S_1 \\
  S_2 & S_2 & S_2 & S_2 \\
  S_1^{-1} & S_1^{-1} & S_1^{-1} & S_1^{-1} \\
  S_2^{-1} & S_2^{-1} & S_2^{-1} & S_2^{-1}
\end{array}
\]

(b) SL in even rounds

\[
\begin{array}{cccc}
  S_1^{-1} & S_1^{-1} & S_1^{-1} & S_1^{-1} \\
  S_2^{-1} & S_2^{-1} & S_2^{-1} & S_2^{-1} \\
  S_1 & S_1 & S_1 & S_1 \\
  S_2 & S_2 & S_2 & S_2
\end{array}
\]

Fig. 2: The two types of SL
The internal state is multiplied by the involution binary matrix with a branch number of eight, so difference propagation over DL has the minimum branch number of eight. Given the input state $x_i$, the output state $y_i$ of the DL operation is computed as follows:

$$
\begin{pmatrix}
    y_0 \\
    y_1 \\
    y_2 \\
    y_3 \\
    y_4 \\
    y_5 \\
    y_6 \\
    y_7 \\
    y_8 \\
    y_9 \\
    y_{10} \\
    y_{11} \\
    y_{12} \\
    y_{13} \\
    y_{14} \\
    y_{15}
\end{pmatrix}
= \begin{pmatrix}
    0001101011000110 \\
    0010010111001001 \\
    0100101000111001 \\
    1000010100110110 \\
    1010010010010011 \\
    0101100001100011 \\
    1010000101101100 \\
    0101001010011100 \\
    1100100100100101 \\
    1100011000110101 \\
    0011011010000101 \\
    0011100101001010 \\
    0110001101011000 \\
    1001001110100100 \\
    1001110001010010 \\
    0110110010100001
\end{pmatrix}
\begin{pmatrix}
    x_0 \\
    x_1 \\
    x_2 \\
    x_3 \\
    x_4 \\
    x_5 \\
    x_6 \\
    x_7 \\
    x_8 \\
    x_9 \\
    x_{10} \\
    x_{11} \\
    x_{12} \\
    x_{13} \\
    x_{14} \\
    x_{15}
\end{pmatrix}
$$

**Key Schedule.** The key schedule algorithm of ARIA takes the master key, $MK$, as input and outputs 13, 15, or 17 128-bit round keys for ARIA-128, ARIA-192, and ARIA-256, respectively. First, $MK$ is divided into two 128-bit values: $KL$ and $KR$. $KL$ is the leftmost 128-bits of $MK$, and $KR$ is the remaining bits; if necessary, all or part of $KR$ is right-padded with zeros. Then, using a 3-round 256-bit Feistel structure, $W_0, W_1, W_2,$ and $W_3$ are generated from $MK$ as follows:

$$
W_0 = KL, \quad W_1 = F_o(W_0, CK_1) \oplus KR, \\
W_2 = F_e(W_1, CK_2) \oplus W_0, \quad W_3 = F_o(W_2, CK_3) \oplus W_1,
$$

where $F_o$ and $F_e$, respectively, denote the odd and even round functions of ARIA, replacing the RKA operation with predefined constants ($CK_1, CK_2,$ and $CK_3$) addition. The key schedule algorithm is approximated by $16 \times 3 = 48$ S-box computations. The 17 round keys are generated from $W_0, W_1, W_2,$ and $W_3$ as follows:
We briefly describe the PGV [31] compression functions of the SBL hash functions (DM, MMD, and MP) and the compression functions of the DBL hash functions (Hirose [13] and MJH [24]). The PGV models, proposed by Preneel et al. in 1993, are typical SBL hash functions. They originally considered 64 block cipher-based hash functions. Subsequently, 12 of these models were demonstrated to be provably secure [1]. The Hirose compression function was proposed by Hirose, and the MJH compression function was proposed by Lee and Stam; they are also provably secure.

Let $E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$ be an $n$-bit keyed block cipher. SBL hash functions call $E$ once to generate the hash value of message $M_i$, and we denote the chaining variables by $H_i$. DBL hash functions call $E$ twice to generate the hash value of message $M_i$, and we denote the chaining variables by $(G_i, H_i)$. Particularly, for MJH, we define the additionally used function $\theta$ as $\theta(x) := k \cdot x$, where $k$ is a nonzero constant and $\cdot$ indicates a multiplication in $\mathbb{F}_2^n$, and we divide $M_i$ into $M_i^1$ and $M_i^2 (M_i = M_i^1 \| M_i^2)$. The involution function $\sigma$ that is commonly used in the Hirose and MJH compression functions is defined as $\sigma(x) := x \oplus c$, where $c$ is a nonzero constant. $\sigma$ is the nonfixed point involution function. The $i^{th}$ compression functions of the SBL and DBL hash functions are described in Figure 3.

### 2.3 Quantum Computation

We use the standard quantum circuit model as the quantum computation model, and adopt \{H, CNOT, T\} (Clifford+T gates) as a basic set of quantum gates [27]. $H$ is the single qubit Hadamard gate defined by $H : |b\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle)$, CNOT is the two-qubit controlled NOT gate defined by $CNOT : |a\rangle|b\rangle \mapsto |a\rangle|b \oplus a\rangle$, and $T$ is the single qubit $\pi/8$ gate defined by $T : |0\rangle \mapsto |0\rangle$ and $T : |1\rangle \mapsto e^{\frac{\pi}{8}}|1\rangle$. We denote the identity operator on $n$-qubit states as $I_n$. 

\[
\begin{align*}
  k_1 &= (W_0) \oplus (W_1 >>> 19), & k_2 &= (W_1) \oplus (W_2 >>> 19), \\
  k_3 &= (W_2) \oplus (W_3 >>> 19), & k_4 &= (W_0 >>> 19) \oplus (W_3), \\
  k_5 &= (W_0) \oplus (W_1 >>> 31), & k_6 &= (W_1) \oplus (W_2 >>> 31), \\
  k_7 &= (W_2) \oplus (W_3 >>> 31), & k_8 &= (W_0 >>> 31) \oplus (W_3), \\
  k_9 &= (W_0) \oplus (W_1 >>> 61), & k_{10} &= (W_1) \oplus (W_2 <<< 61), \\
  k_{11} &= (W_2) \oplus (W_3 >>> 61), & k_{12} &= (W_0 <<< 61) \oplus (W_3), \\
  k_{13} &= (W_0) \oplus (W_1 <<< 31), & k_{14} &= (W_1) \oplus (W_2 <<< 31), \\
  k_{15} &= (W_2) \oplus (W_3 <<< 31), & k_{16} &= (W_0 <<< 31) \oplus (W_3), \\
  k_{17} &= (W_0) \oplus (W_1 <<< 19).
\end{align*}
\]
Quantum Oracle. Consider a Boolean function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \). The quantum oracle of \( f \) is modeled by the unitary operator \( U_f \), which is defined as \( U_f : |x\rangle|q\rangle \mapsto |x\rangle|q \oplus f(x)\rangle \), where \( x \in \{0, 1\}^n \) and \( q \in \{0, 1\} \). \( U_f \) works on \((n + 1)\)-qubits, and the oracle qubit \(|q\rangle\) is flipped when \( f(x) = 1 \); otherwise, it is unchanged. If there is an efficient reversible classical circuit that computes \( f \), \( U_f \) can also be efficiently implemented in a quantum circuit. To construct the quantum oracle \( U_f \), we first construct an efficient reversible classical circuit of \( f \) and substitute it with quantum gates. This makes it possible to uncompute temporary qubits after use.

Grover’s Algorithm. Grover’s algorithm [11] is a quantum search algorithm that can provide a quadratic speedup over brute force when finding desired data from an unstructured database. Consider the following problem:

**Problem 1.** Let \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) be a Boolean function such that \( v := |f^{-1}(1)| > 0 \), and let \( f \) be a black-box. Find \( x \) such that \( f(x) = 1 \).

We define the probability of obtaining the solution \( x \) as \( p := v/2^n \). In a classical setting, we have to make \( O(1/p) \) classical queries to find an \( x \) that satisfies \( f(x) = 1 \). In a quantum setting, we apply Grover’s algorithm to find the solution \( x \) by making only \( O(\sqrt{1/p}) \) quantum queries. That is, Grover’s algorithm used on quantum computers achieves a quadratic speedup, unlike classical algorithms.

To explain in more detail, assume that there is a quantum circuit that performs the quantum oracle \( U_f \) in time \( T_{U_f} \). Then, Grover’s algorithm finds \( x \) in time \( T_{U_f} \cdot (\pi/4) \cdot \sqrt{1/p} \). Grover’s algorithm on a function \( f \) runs the following procedure:

---

**Fig. 3:** The \( i \)th compression functions of the SBL and DBL hash functions.
1. Using the Hadamard gates, prepare the following initial state.

\[ |\psi_{\text{init}}\rangle := H^{\otimes(n+1)}|0^n\rangle|1\].

2. Set \( \theta \) to a value that satisfies \( \sin^2\theta = p \) and \( 0 \leq \theta \leq \pi/2 \). After setting \( i := \lfloor \pi/4 \theta \rfloor \), define \( D_f := (H^{\otimes n} \otimes I_1)(O_0 \otimes I_1)(H^{\otimes n} \otimes I_1) \) as the diffusion operator. Here, \( O_0|0\rangle = (-1)^{\Delta_{x,y}}|x\rangle \) holds, where \( \Delta_{x,y} \) is the Kronecker delta satisfying \( \Delta_{x,y} = 1 \) if \( x = y \); otherwise, \( \Delta_{x,y} = 0 \). Then, perform the unitary operator \( O_f := -D_f U_f \) iteratively \( i \) times for \( |\psi_{\text{init}}\rangle \). We define \( O_f \) as the Grover operator of \( f \).

3. Measure the resulting state of \((O_f)^i|\psi_{\text{init}}\rangle\) and output the most significant \( n \) bits.

In Step 2, if the Grover operator \( O_f \) is repeatedly applied to \( |\psi_{\text{init}}\rangle \), the amplitude of the solution \( x \) is increased. To measure the exact complexity of Grover’s algorithm, we need to accurately measure the complexity of \( U_f \). We elaborate on this analysis later.

The authors of \cite{3} found that when the number of iterations of Grover’s algorithm, \( i \), is set to \( \lfloor \pi/4 \theta \rfloor \), the probability of finding \( x \) such that \( f(x) = 1 \) is at least \( 1 - p \). In addition, we could consider the parallelization of Grover’s algorithm. When the size of \( U_f \) is \( S_f \), and \( S(\geq S_f) \) quantum computers are available, each computer can execute Grover’s algorithm in parallel, where the number of iterations of Grover’s algorithm is \( \lfloor \pi/4 \theta \sqrt{S/S_f} \rfloor \). Then, we can find the solution in time \( T_{U_f} \cdot (\pi/4) \cdot \sqrt{S_f/p} \) with a probability of at least \( 1 - 1/e \approx 0.63 \) \cite{16}.

### 2.4 Dedicated Quantum Collision Attacks

Following Hosoyamada and Sasaki’s dedicated quantum collision attacks on AES hashing modes \cite{15}, further dedicated quantum attacks on AES hashing modes \cite{9}, Hirose \cite{6}, Gimli \cite{12}, SHA-2 \cite{16}, and Simpira v2 \cite{26} were proposed. These attacks showed that an attacker with access to quantum computers can break more rounds of hash functions than one using only classical computers. In a classical setting, the generic attack complexity of finding collisions of an \( n \)-bit ideal hash function is \( O(2^{n/2}) \) by the birthday paradox. In a quantum setting, the generic attack complexity of finding collisions depends on the resources that the attacker can access, and the cryptology community is currently considering the following three quantum settings:

- The attacker can use a polynomially small quantum computer and an exponentially large qRAM.
- The attacker can use a polynomially small quantum computer and an exponentially large classical memory.
- The efficiency of the attacker’s quantum algorithms is evaluated by their time–space tradeoff.
Quantum Rebound Attacks on Reduced-Round ARIA-Based Hash Functions

In the first setting, the best quantum collision finding algorithm is the BHT algorithm proposed by Brassard, Høyer, and Tapp [4]. This algorithm finds collisions in time $O(2^{n/3})$ when a qRAM of $O(2^{n/3})$ is available. In the second setting, the best quantum collision finding algorithm is the CNS algorithm proposed by Chailloux, Naya-Plasencia, and Schrottenloher [5]. This algorithm finds collisions in time $O(2^{2n/5})$ when a quantum computer of $O(n)$ and a classical memory of $O(2^{n/5})$ are available. Our attacks focus on the third quantum setting, and we do not consider qubit communication costs and quantum error corrections.

Time–Space Tradeoff as a Cost Metric. This setting measures attack efficiency as a tradeoff between $T$ and $S$, where $T$ is the attack time complexity and $S$ is the size of the hardware required for the attack. For a quantum attack, $S$ is the size of the quantum computers. Roughly speaking, when a classical computer of size $S$ is available, we can find collisions of a random function in time $T = O(2^{n/2}/S)$ using the parallel rho method [29]. This algorithm was initially proposed for classical computers, but it contains no logical flaws when it is applied to quantum computers. Thus, we can also consider the time–space tradeoff metric as the threshold for quantum attacks. If we can construct a quantum attack that satisfies $T \cdot S < 2^{n/2}$, then the attack is valid in the time–space tradeoff metric.

3 Quantum Rebound Attacks on ARIA-Based SBL Hash Functions

In this section, we discuss the core of the rebound attack, review the quantum rebound attack on ARIA-DM in [10], and present our revised quantum rebound attack on ARIA-DM, which can also be applied to ARIA-MMO and ARIA-MP.

3.1 Rebound Attack

The rebound attack is a hash function analysis technique that was first proposed by Mendel et al. [25] to attack reduced-round Whirlpool and Grøstl. The core of this technique is to exploit the available degrees of freedom in an internal state and the truncated differential to fulfill the low probability part of a differential trail. This part is called the inbound phase and is usually located in the middle of the trail, and it is followed by a probabilistic outbound phase. Generally, the differential propagation in a rebound attack is designed to be dense and sparse in the inbound and outbound phases, respectively. Figure 4 shows an overview of this attack. Here, $F$ is an internal block cipher or permutation that is divided into three parts: $F_{bw}$, $F_{in}$, and $F_{fw}$.

In a quantum setting, to perform a rebound attack on a target primitive with quantum computers, we run Grover’s algorithm on a Boolean function $f(\Delta_{in}, \Delta_{out})$, defined as $f(\Delta_{in}, \Delta_{out}) = 1$, if and only if we get message pairs that satisfy the following conditions:
1. For a given differential trail $\Delta_{in} \rightarrow \Delta_{out}$, obtain an input $(M^I_1, M^I_2)$ and output pair $(M^O_1, M^O_2)$ that conform to the trail, where $(M^I_1, M^I_2)$ and $(M^O_1, M^O_2)$ are called starting points.

2. By propagating $(M^I_1, M^I_2)$ and $(M^O_1, M^O_2)$ to the beginning ($F_{bw}$) and end ($F_{fw}$) of the cipher, respectively, check whether the differential transformations of the given differential trail are satisfied.

If the probability of a differential trail that excludes the probability of the inbound phase is $p$, we must produce $1/p$ starting points, so that at least one pair follows the differential trail for the outbound phase. For this approach to work, the available degrees of freedom should be larger than $1/p$.

3.2 The Quantum Rebound Attack of Dou et al. on ARIA-DM

Dou et al. proposed a quantum rebound attack on ARIA-DM that covers 7 rounds. They used the degrees of freedom in the states for the attack and calculated the probability of finding collisions considering the feed-forward operation. The algorithm of the quantum rebound attack, as described in [10], is as follows (Figure 5):

1. For each of the $2^{56}$ values of $\Delta Y_3$ and $\Delta Z_4$, find the actual pairs of $Y_3$ and $Z_4$ by applying Grover’s algorithm.

2. For the desired differences $\Delta X_3$ and $\Delta Y_5$, check whether $SL^{-1}(Y_3) \oplus SL^{-1}(Y_3 \oplus \Delta Y_3) = \Delta X_3$ holds for $Y_3$ and $\Delta Y_3$, and whether $SL(Z_4 \oplus k_5) \oplus SL(Z_4 \oplus \Delta Z_4 \oplus k_5) = \Delta Y_5$ holds for $Z_4$ and $\Delta Z_4$.

3. After propagating $(X_3, X_3 \oplus \Delta X_3)$ and $(Y_5, Y_5 \oplus \Delta Y_5)$ to the beginning ($F_{bw}$) and end ($F_{fw}$) of the cipher, check whether the difference cancellation occurs in the feed-forward operation.

On the Implausibility of Dou et al.’s Attack. There are three issues that arise in Dou et al.’s attack. First, the complexity of finding collisions is inferior to the cost metric of any quantum setting that is currently considered. According to [10], the probabilities of satisfying Steps 2 and 3 are about $2^{-112}$ and $2^{-56}$, respectively. Thus, even if the complexity of the inbound phase is not considered,
Fig. 5: The quantum rebound attack on 7-round ARIA-DM by Dou et al.

The white and gray boxes denote zero and nonzero differences, respectively.
the complexity of finding collisions is about $2^{84} = \sqrt{2^{112 \times 80}}$, which is inferior to the generic attack complexity of the three quantum settings in Section 2. In particular, the probability of $2^{-112}$ in Step 2 can be improved to $2^{-96}$, but this improvement does not make their attack faster than generic attacks in quantum settings. Second, the calculation of the available degrees of freedom they performed was incorrect. The degrees of freedom in the states that would be required to perform their attack are greater than $2^{168}$, although they insisted that $2^{168}$ degrees of freedom could be obtained from the triple $(\Delta Y_3, \Delta Z_4, \Delta X_5)$. However, considering the differential trail, if $\Delta Z_4$ is determined, $\Delta X_5$ is determined accordingly. Therefore, there are only $2^{112}$ degrees of freedom that could be obtained from the triple $(\Delta Y_3, \Delta Z_4, \Delta X_5)$, and in fact, the attack is invalid. Third, the calculation of the inbound phase was not accurately described. The authors argued that by applying Grover’s algorithm in Step 1, they could find compatible starting points $Y_3$ and $Z_4$ for $\Delta Y_3$ and $\Delta Z_4$, respectively. However, they did not describe the detailed process of finding the starting points using Grover’s algorithm and calculating the required complexity. For a more accurate and improved complexity estimation, we could apply Grover’s algorithm to each S-box to find a matching starting point, but this approach that is not considered in this study. For a precise cryptanalysis, all of these issues should be resolved.

### 3.3 Our Revised Attack on 5-Round ARIA-DM, ARIA-MMO, and ARIA-MP

In this section, we describe our revised quantum rebound attacks on 5-round ARIA-DM, ARIA-MMO, and ARIA-MP, which are valid for all key lengths of ARIA. We performed a thorough analysis and found that a quantum attack that is superior to the generic attack complexity of a quantum setting could be constructed up to 5 rounds, but not 7 rounds. Our attacks are faster than the generic attack in the cost metric of time–space tradeoff and slower than the generic attacks in other quantum settings. For ARIA-DM, our quantum rebound attack is used to find free-start collisions, whereas for ARIA-MMO and ARIA-MP, it finds collisions. Since all attack processes are equally applied to the three structures, we focus on ARIA-DM.

#### Implementation of $f$.

The core of our attack is to force the element of interest in our search space to stand out among the other entries by applying Grover’s algorithm. We denote the input–output difference pair of the inbound phase (Figure 6) by $(\Delta_{in}, \Delta_{out})$, where $\Delta_{in} = \Delta Y_2$ and $\Delta_{out} = \Delta Z_3$. Since the attack requires $2^{104}$ degrees of freedom, we consider $\Delta_{out}$ as an element of $\mathbb{F}_2^{48}$. First, we define a Boolean function:

$$ f : \mathbb{F}_2^{56} \times \mathbb{F}_2^{48} \to \mathbb{F}_2 $$

where $f(\Delta_{in}, \Delta_{out}) = 1$ holds if and only if the starting point computed with $(\Delta_{in}, \Delta_{out})$ satisfies the following conditions:

1. The starting point $(X_3, X_3 \oplus \Delta X_3)$ satisfies the differential transformations of part $F_{bw}$. 
2. The starting point \((X_3, X_3 \oplus \Delta X_3)\) satisfies the differential transformations of part \(F_{f_{iw}}\).

If \(f(\Delta_{in}, \Delta_{out}) = 1\) holds, we can compute an input pair \((H_0, H_0')\) that produces collisions. We only use a fraction of the degrees of freedom that \(\Delta_{out}\) has and, on average, expect that there is one starting point \((X_3, X_3 \oplus \Delta X_3)\) for each \((\Delta_{in}, \Delta_{out})\).
For a given $(\Delta_{in}, \Delta_{out})$, the function $f(\Delta_{in}, \Delta_{out})$ can be computed using a classical computer as follows:

1. Compute the differences $(\Delta X^i_3, \Delta Y^i_3)$ ($0 \leq i < 16$) from $(\Delta Y_2, \Delta Z_3)$ in round 3, where $\Delta Y_2 = \Delta_{in}$ and $\Delta Z_3 = \Delta_{out}$.
2. Given the obtained differences, solve the following equation and find one $X^3_i$ on average for each active S-box $S^3_i$ ($0 \leq i < 16$):

$$S^3_i(X^3_i) \oplus S^3_i(X^3_i + \Delta X^3_i) = \Delta Y^3_i. \quad (2)$$

Then, set $X^3_0 = \min\{X^0_3, X^0_3 \oplus \Delta X^0_3\}$, and similarly set $X^3_1, X^3_2, \ldots, X^3_{15}$. With this process, the starting point $X_3 = (X^3_0, X^3_1, \ldots, X^3_{15})$ is constructed. If there are no admissible values for the pair $(\Delta X_3, \Delta Y_3)$, then return to Step 1.

3. Propagate the starting point $(X_3, X_3 \oplus \Delta X_3)$ obtained in Step 2 to the beginning and end of the cipher, and check whether the differential transformations of the differential trail are satisfied. If so, $f(\Delta_{in}, \Delta_{out})$ returns 1; otherwise, it returns 0.

In the $F_{bw}$ process of Step 3, for $Y_1$ to be active only in the 0th byte, the differences of all active bytes of $X_2 (= SL^{-1}(Y_2))$ must be the same. In [10], this probability is calculated as $2^{-56}$, but it should be corrected to $2^{(-8) \times (2^{56} - 1)} \approx 2^{-48}$ because there are $2^8 - 1$ differences that can be equal. This point is equally applied in the $F_{fw}$, and the probability of the outbound phase is $2^{-104}$ considering the feed-forward operation. This is why we set the degrees of freedom to $2^{104}$ in this attack. By applying Grover’s algorithm to the quantum oracle $U_f$, which maps $|\Delta_{in}, \Delta_{out}\rangle|q\rangle$ to $|\Delta_{in}, \Delta_{out}\rangle|q\rangle \oplus f(\Delta_{in}, \Delta_{out})\rangle$, we can find collisions with about $T_{UF} \cdot (\pi/4) \cdot \sqrt{2^{104}}$ queries, where $T_{UF}$ is the time required to run the quantum oracle $U_f$. To estimate the overall complexity, we need to clarify the complexity at which $U_f$ runs.

**Implementation of the Quantum Oracle $U_f$.** Below, we describe how to implement $f$ on quantum computers, or equivalently, how to implement the unitary operator $U_f$, defined as $U_f : |\Delta_{in}, \Delta_{out}\rangle|q\rangle \mapsto |\Delta_{in}, \Delta_{out}\rangle|q\rangle \oplus f(\Delta_{in}, \Delta_{out})\rangle$. As in the studies [15,26,9,16], we need to define an additional function $G^i$ to implement $U_f$. $G^i$ finds, on average, one actual input value that satisfies the input–output difference pair of each S-box $S^i$ ($0 \leq i < 16$). Specifically, $G^i$ outputs $X^i_3 = \min\{X^i_3, X^i_3 \oplus \Delta X^i_3\}$ that satisfies $S^3_i(X^i_3) \oplus S^3_i(X^i_3 \oplus \Delta X^i_3) = \Delta Y^i_3$ concerning the input–output difference pair $(\Delta X^i_3, \Delta Y^i_3)$ in round 3. We eliminate the requirement of qRAM to implement a differential distribution table by applying Grover’s algorithm to $G^i$. The implementation of the quantum oracle $U_{G^i}$ is presented in Algorithm 1. Finally, the implementation of the quantum oracle $U_f$ is presented in Algorithm 2.

**Complexity Analysis.** To analyze the complexity of finding collisions, the following should be considered:
Algorithm 1: Implementation of $U_G$

**Input:** $|\Delta X_i^3, \Delta Y_i^3; X_i^3\rangle|q\rangle$

**Output:** $|\Delta X_i^3, \Delta Y_i^3; X_i^3\rangle|q\rangle \oplus G_i(\Delta X_i^3, \Delta Y_i^3; X_i^3)$

1. Set $X_i^3 \leftarrow \min(X_i^3, X_i^3 \oplus \Delta X_i^3)$.
2. If $S_i^3(X_i^3) \oplus S_i^3(X_i^3 \oplus \Delta X_i^3) = \Delta Y_i^3$ then
   3. Return $|\Delta X_i^3, \Delta Y_i^3; X_i^3\rangle|q\rangle \oplus 1\rangle$
   4. Else
   5. Return $|\Delta X_i^3, \Delta Y_i^3; X_i^3\rangle|q\rangle$
6. End

Algorithm 2: Implementation of $U_f$

**Input:** $|\Delta_{in}, \Delta_{out}\rangle|q\rangle$

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

1. /* inbound phase */
2. For $i \in \{0, 1, \ldots, 15\}$ do
3.   Compute the differences $(\Delta X_i^3, \Delta Y_i^3)$ from $(\Delta Y_2, \Delta Z_3)$, where $\Delta Y_2 = \Delta_{in}$ and $\Delta Z_3 = \Delta_{out}$.
4.   Run $G^i(\Delta X_i^3, \Delta Y_i^3; X_i^3)$. Let $(X_i^3, X_i^3 \oplus \Delta X_i^3)$ be the output.
5. End
6. Set $X_3 \leftarrow (X_3^0, \ldots, X_3^{15})$ and $X'_3 \leftarrow (X_3^0 \oplus \Delta X_3^0, \ldots, X_3^{15} \oplus \Delta X_3^{15})$.
7. /* outbound phase */
8. If $(X_3, X'_3)$ fulfills the differential transformations of the outbound phase then
9.   Return $|\Delta_{in}, \Delta_{out}\rangle|q\rangle \oplus 1\rangle$
10. Else
11.   Return $|\Delta_{in}, \Delta_{out}\rangle|q\rangle$
12. End

The complexity of the computation of 5-round ARIA is approximated by $16 \times (5 + 3) = 128$ S-box computations.

One computation of an inverse S-box is almost the same as the computation of two S-boxes \[17\].

Uncomputations are considered to free up the wires of the quantum circuit after performing $U_f$.

The study presented in \[17\] was originally performed on the S-box of AES. However, $S_1$, the S-box used in ARIA, is the same as that of AES, and $S_2$ is defined similarly to $S_1$ to be an affine transformation of the inversion function over $GF(2^8)$. Thus, we expect the complexity of $S_2^{-1}$ to be almost twice that of $S_2$.

**Complexity of $G^i$.** For a given $(\Delta X_i^3, \Delta Y_i^3)$, to find $X_i^3$ by applying Grover’s algorithm to $G^i$, we need to query $U_G$. For odd rounds, the complexity of $G^i$ depends on $i$, since SL consists of the first two rows as $S_1$ and $S_2$, and the other two rows as inverses of each S-box. Since the number of queries required by Grover’s algorithm is $(\pi/4) \times \sqrt{2^8} \approx 23.65$, when $i = 0, 1, 4, 5, 8, 9, 12, 13$ (corresponds to the 1st
and 2nd rows), the complexity of $G_i$ is equivalent to $2 \times (\pi/4) \times \sqrt{2^8} \times (1/128) \approx 2^{-2.35}$ 5-round ARIA computations, where 128 is the number of S-boxes to which the 5-round ARIA is approximated. If $i = 2, 3, 6, 7, 10, 11, 14, 15$ (corresponds to the 3rd and 4th rows), then the complexity of $G_i$ is equivalent to $2 \times 2 \times (\pi/4) \times \sqrt{2^8} \times (1/128) \approx 2^{-1.35}$ 5-round ARIA computations. Therefore, the total complexity of $G_i$ is $8 \times 2^{-2.35} + 8 \times 2^{-1.35} \approx 2^{2.23}$.

**Complexity of $U_f$.** The implementation of $U_f$ includes 16 calls of $G_i$ in Steps 2–5, which require $8 \times 2^{-2.35} + 8 \times 2^{-1.35} \approx 2^{2.23}$ 5-round ARIA computations. We need to perform S-box computations from the starting point $X_3$ to both ends of the cipher. Since there are half inverse S-boxes in the S-box layer of each round, we need $(8 + 8 \times 2) \times 5 \times 2 \times (1/128) \approx 2^{0.9}$ 5-round ARIA computations. Hence, the overall complexity of $U_f$ is $2 \times (2^{2.23} + 2^{0.9}) \approx 2^{3.71}$ 5-round ARIA computations.

**Overall Complexity of Finding Collisions.** First, the number of qubits (or, the unit of size) required to implement DM instantiated with ARIA-128 is 256. For ARIA-192 and ARIA-256, 320 and 384 qubits are required, respectively. This is the only part that depends on the key length. To estimate $S_f$, we need $2 \times 128$ qubits to store $(\Delta_{in}, \Delta_{out})$ and a single qubit for $q$. Steps 3 and 4 require an additional $(16 \times 8 \times 2 + 8 \times 2) = 272$ qubits to run $G_i$ as well as to compute and store the values of input–output difference pairs. Step 6 requires an additional $2 \times 128$ qubits to store $X_3$ and $X'_3$. Step 8 requires an additional $128 \times 5 = 640$ qubits. Thus, to store all values shown in the above implementation, 1425 qubits are used in total. Hence, we have

$$S_f \leq 1425/256 \leq 2^{2.48}.$$  

If we consider the parallelization of Grover’s algorithm when $S(\geq 2^{2.48})$ quantum computers are available, our rebound attacks run in time $(\pi/4) \times 2^{2.71} \times \sqrt{2^{2.48}/(2^{-104} \cdot S)} \leq 2^{56.61}/\sqrt{S}$. Our attacks are faster than the generic attack complexity $2^{64}/S$ in the cost metric of time–space tradeoff, as long as $2^{2.48} \leq S < 2^{14.78}$. $S_f$ is $2^{2.15}$ and $2^{1.89}$ in the case of ARIA-192 and ARIA-256, respectively, so even if the key size increases, the attack complexity does not differ by more than a factor of $2^{0.5}$.

**Remark 1.** Considering the structure of the compression function, this attack is mounted as a free-start collision attack for ARIA-DM and as a collision attack for ARIA-MMO and ARIA-MP.

### 4 Quantum Rebound Attacks on ARIA-Based DBL Hash Functions

In this section, we provide a new differential trail for 7-round ARIA and mount it on ARIA-Hirose and ARIA-MJH to perform our quantum rebound attacks. We
adopt the strategy of finding collisions that satisfy condition $\Delta G_0 = G_0 \oplus G'_0 = c$ in the entire compression function, where $c$ has one nonzero byte at the 0th position. Since all attack processes are equally applied to the two structures, we focus on ARIA-Hirose.

4.1 New Differential Trail for 7-Round ARIA

As in [23], we propose a differential trail for 7-round ARIA with a probability of $2^{-112}$ using the degrees of freedom from the key schedule.

New Differential Trail Using Two Inbound Phases. We construct a trail by setting up two inbound phases and connecting them using the connection phase (Figure 7). The two inbound phases are performed using the same process as presented in Section 3, and the core of our attack is to thoroughly analyze the connection phase. First, we set inbound phases 1 and 2 to be placed on rounds 2.5–3 and 4.5–5, respectively, and we compute the starting points of $X_3$ and $X_5$. Due to the nature of hash functions as keyless primitives, an attacker can choose a message pair that satisfies a given differential trail. From this perspective, the degrees of freedom obtainable from the key schedule are used to connect the starting points of $X_3$ and $X_5$. Finally, $\Delta X_0$ and $\Delta X_8$ can be computed by propagating the starting points ($X_3$, $X_3 \oplus \Delta X_3$) and ($X_5$, $X_5 \oplus \Delta X_5$) to the beginning and end of the cipher, respectively. Since the probability of canceling one byte for the feed-forward operation is $2^{-8}$ and $\Delta G_0 = c$ must hold, the overall time complexity of the attack is $2^{96} \times 2^8 \times 2^8 = 2^{112}$.

Connecting Two Inbound Phases. Our overall connection process is shown in Figure 8. To connect the results of two inbound phases, we perform an exhaustive search on $W_0 = KL$ in the key schedule, which is a search that has the highest complexity in this attack. Recall that $K_4 = (W_0 >>> 19) \oplus (W_3)$ and $K_5 = (W_0) \oplus (W_1 >>> 31)$ hold. In the key schedule, $Y_4$ and $X_5$ can be connected by appropriately adjusting $W_1 = F_o(W_0) \oplus KR$ according to the fixed $W_0$. Since $k_4$ is determined according to $k_5$, the connection between $X_3$ and $Z_3$ must be approached probabilistically. The detailed calculation process is as follows:

1. Compute $Z_3$ and $DL(X_5)$ from starting points $X_3$ and $X_5$.
2. For the input–output difference pair ($\Delta X_4$, $\Delta Y_4$), obtain $X_4$ that is compatible with the pair.
3. Fix the value of $W_0$, and determine $W_1$ so that the given values $Y_4$ and $DL(X_5)$ can be connected. Here, because $k_5$ is determined, $k_4$ is also determined.
4. Given the value of $k_4$, check whether $Z_3 \oplus k_4 = X_4$ holds. If not, repeat from Step 2.

By performing this process, we can find round keys $k_4$ and $k_5$ that connect $X_3$ and $X_5$. Notably, $\Delta X_4 = \Delta Z_3$ and $\Delta Y_4 = \Delta DL(X_5)$ hold, and the $\Delta X_4$ and $\Delta Y_4$ differences are well connected.
Fig. 7: Our quantum rebound attack on 7-round ARIA-Hirose
4.2 Quantum Collision Attacks on 7-Round ARIA-Hirose and ARIA-MJH

In this section, we describe our quantum rebound attacks on 7-round ARIA-Hirose and ARIA-MJH; these attacks are valid for the 256-bit key length of ARIA.

**Implementation of \( f \).** The core of our attack is the same as in Section 3. For two inbound phases, we denote the input–output difference pair by \((\Delta_{\text{in}}, \Delta_{\text{out}}) = (\Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_1^{\text{out}}, \Delta_2^{\text{out}})\), where \((\Delta_1^{\text{in}}, \Delta_1^{\text{out}})\) is the input–output difference pair of the first inbound phase and \((\Delta_2^{\text{in}}, \Delta_2^{\text{out}})\) is that of the second inbound phase. Since the attack requires \(2^{112}\) degrees of freedom, we consider \(\Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_1^{\text{out}}, \Delta_2^{\text{out}}\) as elements of \(\mathbb{F}_2^{28}\), respectively. First, we define a Boolean function:

\[
\begin{align*}
\mathbb{F}_2^{28} \times \mathbb{F}_2^{28} \times \mathbb{F}_2^{28} \times \mathbb{F}_2^{28} \rightarrow \mathbb{F}_2
\end{align*}
\]

where \(f(\Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_1^{\text{out}}, \Delta_2^{\text{out}}) = 1\) holds if and only if the starting points computed with \((\Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_1^{\text{out}}, \Delta_2^{\text{out}})\) satisfy the following conditions:

1. The starting point \((X_3, X_3 \oplus \Delta X_3)\) satisfies the differential transformations of part \(F_{bw}\).
2. The starting point \((X_5, X_5 \oplus \Delta X_5)\) satisfies the differential transformations of part \(F_{fw}\).

If \(f(\Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_1^{\text{out}}, \Delta_2^{\text{out}}) = 1\) holds, we can compute the input values that make collisions. We use only a fraction of the degrees of freedom that \((\Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_1^{\text{out}}, \Delta_2^{\text{out}})\) has and, on average, expect that there is one pair of starting points \((X_3, X_3 \oplus \Delta X_3)\) and \((X_5, X_5 \oplus \Delta X_5)\) for each \((\Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_1^{\text{out}}, \Delta_2^{\text{out}})\).

For a given \((\Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_1^{\text{out}}, \Delta_2^{\text{out}})\), the function \(f(\Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_1^{\text{out}}, \Delta_2^{\text{out}})\) can be computed using a classical computer as follows:

1. Choose nine random values in \(\{0, 1\}^8\) for the state to be constructed later.
2. Compute the differences \((\Delta X_3^i, \Delta Y_3^i)\) \((0 \leq i < 16)\) from \((\Delta Y_2, \Delta Z_3)\) in round 3, where \(\Delta Y_2 = \Delta_1^{\text{in}}\) and \(\Delta Z_3 = \Delta_1^{\text{out}}\).
3. Given the obtained differences, solve the following equation and find one \( X_3^i \) on average for each active S-box \( S_3^i \) (0 \( \leq i \) \( < 16 \)):

\[
S_3^i(X_3^0) \oplus S_3^i(X_3^0 \oplus \Delta X_3^0) = \Delta Y_3^i.
\]

Then, set \( X_3^0 = \min\{X_3^0, X_3^0 \oplus \Delta X_3^0\} \), and similarly set \( X_3^1, X_3^2, \ldots, X_3^{15} \).

With this process, the starting point \( X_3 = (X_3^0, X_3^1, \ldots, X_3^{15}) \) is constructed.

If there are no admissible values for the pair \( (\Delta X_3, \Delta Y_3) \), then return to Step 2.

4. Compute the differences \( (\Delta X_3^i, \Delta Y_3^i) \) (0 \( \leq i \) \( < 16 \)) from \( (\Delta Y_4, \Delta Z_5) \) in round 5, where \( \Delta Y_4 = \Delta^2_{in} \) and \( \Delta Z_5 = \Delta^2_{out} \).

5. Given the obtained differences, solve the following equation and find one \( X_5^i \) on average for each active S-box \( S_5^i \) (0 \( \leq i \) \( < 16 \)):

\[
S_5^i(X_5^0) \oplus S_5^i(X_5^0 \oplus \Delta X_5^0) = \Delta Y_5^i.
\]

Then, set \( X_5^0 = \min\{X_5^0, X_5^0 \oplus \Delta X_5^0\} \), and similarly set \( X_5^1, X_5^2, \ldots, X_5^{15} \).

With this process, the starting point \( X_5 = (X_5^0, X_5^1, \ldots, X_5^{15}) \) is constructed.

If there are no admissible values for the pair \( (\Delta X_5, \Delta Y_5) \), then return to Step 4.

6. Given the difference pair \( (\Delta X_4^i, \Delta Y_4^i) \), solve the following equation and find one \( X_4^i \) on average for each active S-box \( S_4^i \) (i = 3, 4, 6, 8, 9, 13, 14):

\[
S_4^i(X_4^0) \oplus S_4^i(X_4^0 \oplus \Delta X_4^0) = \Delta Y_4^i.
\]

The seven corresponding values of \( Y_4^i \) are determined by the above solution.

Compute \( Z_4 \) after setting the remaining nine bytes of \( Y_4 \) to the random values chosen in Step 1.

7. Do the following for each \( W_0 \):

   (a) Compute \( k_5 \) that is compatible with \( Z_4 \) and starting point \( X_5 \).

   (b) Compute \( k_3 \) from \( k_5 \) and check whether \( Z_3 \oplus k_4 = X_4 \) holds.

   If there is no admissible value for \( W_0 \), then repeat the process in Step 5.

   Notably, the starting points \( X_3 \) and \( X_5 \) are now connected correctly.

8. Propagate starting points \( (X_3, X_3 \oplus \Delta X_3) \) and \( (X_5, X_5 \oplus \Delta X_5) \) to the beginning and end of the cipher, respectively. If the differential transformations of the differential trail are satisfied, \( f(\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out}) \) returns 1; otherwise it returns 0.

Since we need to consider that the feed-forward operation and \( \Delta G_0 \) and \( c \) must be equal, the degrees of freedom we need in this attack are greater than \( 2^{12} \). By applying Grover’s algorithm to the quantum oracle \( \mathcal{U}_f \), which maps \( |\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out}, q\rangle \) to \( |\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out}, q \oplus f(\Delta^1_{in}, \Delta^2_{in}, \Delta^1_{out}, \Delta^2_{out})\rangle \), we can find collisions with about \( T_{\mathcal{U}_f} \cdot (\pi/4) \cdot \sqrt{2^{112}} \) queries, where \( T_{\mathcal{U}_f} \) is the time required to run the quantum oracle \( \mathcal{U}_f \). To estimate the overall complexity, we need to clarify the complexity at which \( \mathcal{U}_f \) runs.
Algorithm 3: Implementation of $U_K$

Input: $|X_4, X_5, Z_3, Z_4; W_0\rangle |q\rangle$
Output: $|X_4, X_5, Z_3, Z_4; W_0\rangle |q\rangle \oplus K(X_4, X_5, Z_3, Z_4; W_0)$

1. Compute $W_1$ from $Z_4$, $X_5$, and $W_0$.
2. Compute $k_4$ from $W_0$ and $W_1$.
3. if $Z_3 \oplus k_4 = X_4$ then
   4. return $|X_4, X_5, Z_3, Z_4; W_0\rangle |q\rangle \oplus 1\rangle$
else
   6. return $|X_4, X_5, Z_3, Z_4; W_0\rangle |q\rangle$
end

Implementation of the Quantum Oracle $U_f$. Below, we describe how to implement $f$ on quantum computers, or equivalently, how to implement the unitary operator $U_f$, defined as $U_f: |\Delta_{in}^1, \Delta_{out}^1, \Delta_{in}^2, \Delta_{out}^2\rangle |q\rangle \mapsto |\Delta_{in}^1, \Delta_{out}^1, \Delta_{in}^2, \Delta_{out}^2\rangle |q\rangle \oplus f(\Delta_{in}^1, \Delta_{out}^1, \Delta_{in}^2, \Delta_{out}^2)\rangle$. As in the studies \[15,26,9,6\], we need two additional functions to implement $U_f$: $G^i$, as was needed in Section 3, and $K$. $G^i$ is used to find $X_3$ in Section 3; but it is used here to find $X_3$, $X_4$, and $X_5$. Next, we define the function $K$, which connects the two starting points $X_3$ and $X_5$. That is, $K$ outputs $W_0$ connecting starting points $X_3$ and $X_5$, from which $k_3$ can be computed. The implementation of the quantum oracle $U_K$ is presented in Algorithm 3. Finally, the implementation of the quantum oracle $U_f$ is presented in Algorithm 3. Note that the preselected values used in Step 16 are chosen in a classical way before running the quantum algorithm.

Complexity Analysis. The complexity of the computation of 7-round ARIA is approximated by $16 \times (7 + 3) = 160$ S-box computations, and the other considerations are the same as those in Section 3.

Complexity of $K$. To find $W_0$ and $W_1$ connecting starting points $X_3$ and $X_5$ by applying Grover’s algorithm to $K$, we need to query to $U_K$. The number of queries required is $(\pi/4) \times \sqrt{2^{128}} \approx 2^{63.65}$, which is equivalent to $(\pi/4) \times \sqrt{2^{128}} \times (48/160) \approx 2^{61.91}$ 7-round ARIA computations.

Complexity of $U_f$. The complexity of $G^i$ is evaluated as $2 \times (\pi/4) \times \sqrt{2^8} \times (1/160) \approx 2^{-2.67}$ 7-round ARIA computations for $i = 0, 1, 4, 5, 8, 9, 12, 13$; otherwise $2 \times 2 \times (\pi/4) \times \sqrt{2^8} \times (1/160) \approx 2^{-1.67}$. The implementation of $U_f$ includes 39 calls of $G^i$ in Steps 2–5, 7–10, and 12–14, which require $20 \times 2^{-2.67} + 19 \times 2^{-1.67} \approx 2^{3.19}$ 7-round ARIA computations. We need to perform S-box computations from starting points $X_3$ and $X_5$ to both ends of the cipher. Since there are half inverse S-boxes in the S-box layer of each round, we need $(8 + 8 \times 2) \times 5 \times 2 \times (1/160) \approx 2^{0.58}$ 7-round ARIA computations. The implementation of $K$ in Step 17 is also included. Hence, the overall complexity of $U_f$ is $2 \times (2^{3.19} + 2^{0.58} + 2^{61.91}) \approx 2^{62.91}$ 7-round ARIA computations.
For the estimation of the unit of size required to implement Hirose are used in total. Hence, we have:

Thus, to store all the values shown in the above implementation, 2977 qubits to store \((\times G)\) difference pairs. Steps 8 and 9 also require 272 qubits in the same manner, and 272 qubits to run \(G\) and a single qubit for \(f\). First, the number of qubits (or, the unit of size) required to implement Hirose instantiated with ARIA-256 is 512. For the estimation of the \(S_f\), we need \(4 \times 128\) qubits to store \((\Delta_{in}^1, \Delta_{in}^2, \Delta_{out}^1, \Delta_{out}^2)\) and a single qubit for \(q\). Steps 3 and 4 require an additional \((16 \times 8 \times 2 + 8 \times 2) = 272\) qubits to run \(G^i\) as well as compute and store the values of input-output difference pairs. Steps 8 and 9 also require 272 qubits in the same manner, and since Step 13 runs \(G^i\) for only seven bytes, 128 qubits are required. Step 17 requires an additional \(128 \times 5\) qubits, and Steps 18 and 19 require \(4 \times 128\) qubits to store \((X_3, X'_3, X_5, X'_5)\). Step 21 requires an additional \(128 \times 5 = 640\) qubits. Thus, to store all the values shown in the above implementation, 2977 qubits are used in total. Hence, we have:

\[
S_f \leq 2977/512 \leq 2^{2.54}.
\]

**Algorithm 4: Implementation of \(U_f\)**

| Input | \(|\Delta_{in}^1, \Delta_{in}^2, \Delta_{out}^1, \Delta_{out}^2\rangle|q\) |
| Output | \(|\Delta_{in}^1, \Delta_{in}^2, \Delta_{out}^1, \Delta_{out}^2\rangle|q \oplus f(\Delta_{in}^1, \Delta_{in}^2, \Delta_{out}^1, \Delta_{out}^2)\) |

1 /* inbound phase 1 */
2 for \(i \in \{0, 1, \ldots, 15\}\) do
3 Compute the differences \((\Delta X_4^i, \Delta Y_4^i)\) from \((\Delta Y_2, \Delta Z_3)\), where \(\Delta Y_2 = \Delta_{in}^1\) and \(\Delta Z_3 = \Delta_{out}^1\).
4 Run \(G^i(\Delta X_3^i, \Delta Y_3^i; X_4^i)\). Let \((X_4^i, X'_4^i \oplus \Delta X_4^i)\) be the output.
5 end
6 /* inbound phase 2 */
7 for \(i \in \{0, 1, \ldots, 15\}\) do
8 Compute the differences \((\Delta X_4^i, \Delta Y_4^i)\) from \((\Delta Y_4, \Delta Z_5)\), where \(\Delta Y_4 = \Delta_{in}^2\) and \(\Delta Z_5 = \Delta_{out}^2\).
9 Run \(G^i(\Delta X_3^i, \Delta Y_3^i; X_4^i)\). Let \((X_4^i, X'_4^i \oplus \Delta X_4^i)\) be the output.
10 end
11 /* connection phase */
12 for \(i \in \{3, 4, 6, 8, 9, 13, 14\}\) do
13 Run \(G^i(\Delta X_4^i, \Delta Y_4^i; X_4^i)\). Let \((X_4^i, X'_4^i \oplus \Delta X_4^i)\) be the output.
14 end
15 Compute the seven corresponding bytes of \(Y_4\).
16 Set the remaining nine bytes of \(Y_4\) to preselected values and compute \(Z_4\).
17 Run \(K(X_4, X_5, Z_3, Z_4; W_0)\) for the exhaustive search of \(W_0\).
18 Set \(X_3 \leftarrow (X_3^0, \ldots, X_3^{15})\) and \(X'_3 \leftarrow (X'_3^0 \oplus \Delta X_3^0, \ldots, X'_3^{15} \oplus \Delta X_3^{15})\).
19 Set \(X_5 \leftarrow (X_5^0, \ldots, X_5^{15})\) and \(X'_5 \leftarrow (X'_5^0 \oplus \Delta X_5^0, \ldots, X'_5^{15} \oplus \Delta X_5^{15})\).
20 /* outbound phase */
21 if \((X_3, X'_3, X_5, X'_5)\) fulfills the differential transformations of the outbound phase then
22 return \(|\Delta_{in}^1, \Delta_{in}^2, \Delta_{out}^1, \Delta_{out}^2\rangle|q \oplus 1\)
23 else
24 return \(|\Delta_{in}^1, \Delta_{in}^2, \Delta_{out}^1, \Delta_{out}^2\rangle|q\)
25 end

**Overall Complexity of Finding Collisions.** First, the number of qubits (or, the unit of size) required to implement Hirose instantiated with ARIA-256 is 512. For the estimation of the \(S_f\), we need \(4 \times 128\) qubits to store \((\Delta_{in}^1, \Delta_{in}^2, \Delta_{out}^1, \Delta_{out}^2)\) and a single qubit for \(q\). Steps 3 and 4 require an additional \((16 \times 8 \times 2 + 8 \times 2) = 272\) qubits to run \(G^i\) as well as compute and store the values of input-output difference pairs. Steps 8 and 9 also require 272 qubits in the same manner, and since Step 13 runs \(G^i\) for only seven bytes, 128 qubits are required. Step 17 requires an additional \(128 \times 5\) qubits, and Steps 18 and 19 require \(4 \times 128\) qubits to store \((X_3, X'_3, X_5, X'_5)\). Step 21 requires an additional \(128 \times 5 = 640\) qubits. Thus, to store all the values shown in the above implementation, 2977 qubits are used in total. Hence, we have:

\[
S_f \leq 2977/512 \leq 2^{2.54}.
\]
If we consider the parallelization of Grover’s algorithm when \( S(\geq 2^{2.54}) \) quantum computers are available, our rebound attacks run in time \((\pi/4) \times 2^{62.91} \times \sqrt{2^{2.54}/(2^{112} \cdot S)} \leq 2^{119.83}/\sqrt{S}\). Our attacks are faster than the generic attack complexity \(2^{128}/S\) in the cost metric of time–space tradeoff as long as \(2^{2.54} \leq S < 2^{16.34}\).

For ARIA-MJH, the number of qubits required for implementation is 640. Thus, \(S_f \leq 2^{2.22}\) holds for ARIA-MJH, and the attack runs in time \((\pi/4) \times 2^{62.91} \times \sqrt{2^{2.22}/(2^{112} \cdot S)} \leq 2^{119.67}/\sqrt{S}\) as long as \(2^{2.22} \leq S < 2^{16.66}\).

**Remark 2.** Considering the structure of the compression function, this attack is mounted as a free-start collision attack for ARIA-Hirose and as a semi-free-start collision attack for ARIA-MJH.

5 Conclusions

In this study, we revised the quantum rebound attacks on SBL hash functions instantiated with ARIA proposed by Dou et al. \[10\], and we proposed new quantum rebound attacks on several DBL hash functions instantiated with ARIA. To find collisions of hash functions, a differential trail for 5-round ARIA was mounted for SBL hash functions, including DM, MMO, and MP, and a differential trail for 7-round ARIA-256 using two inbound phases and a connection phase was mounted for DBL hash functions, including Hirose and MJH. In particular, the 7-round differential trail was newly constructed by exploiting the maximum \(2^{128}\) degrees of freedom in the key schedule of ARIA-256. These results are expected to inspire the analysis of hash functions instantiated with other byte-oriented block ciphers. Extending our attacks to more rounds will be an interesting research topic.

**Acknowledgements.** This work was supported as part of Military Crypto Research Center(UD210027XD) funded by Defense Acquisition Program Administration(DAPA) and Agency for Defense Development(ADD).

**References**


