Mind the Composition of Toffoli Gates: Structural Algebraic Distinguishers of ARADI

Emanuele Bellini[0000−0002−2349−0247], Mohamed Rachidi[0009−0008−5279−4902], Raghvendra Rohit[0000−0002−5272−1016], and Sharwan K. Tiwari[0000−0002−9487−0669]

Technology Innovation Institute, Cryptography Research Center, Abu Dhabi, UAE
{name.lastname}@tii.ae
https://www.tii.ae/cryptography

Abstract. This paper reveals a critical flaw in the design of ARADI, a recently proposed low-latency block cipher by NSA researchers – Patricia Greene, Mark Motley, and Bryan Weeks. The weakness exploits the specific composition of Toffoli gates in the round function of ARADI's nonlinear layer, and it allows the extension of a given algebraic distinguisher to one extra round without any change in the data complexity. More precisely, we show that the cube-sum values, though depending on the secret key bits, are always equal in two of the state words. Such a structural property is difficult to obtain by the direct application of division property and has never been seen before in any state-of-the-art block cipher. We call this structural property *weakly-composed-Toffoli gates*, and introduce a theoretical framework which can describe it in general terms. We present algebraic distinguishers that reach 8 out of 16 rounds of ARADI. Most notably, we show that these distinguishers have better data complexities than the division property-based distinguishers for the same number of rounds. We further investigate whether changing the linear layer or the order of composition of Toffoli gates could avoid this property. We give a negative answer to the same and show that it is impossible to prevent this structural property unless the nonlinear layer is re-designed. As a side result, we provide a key-recovery attack on 10 rounds ARADI with 2^{124} data and 2^{177} time for a 256-bit key. Our work highlights the significance of security analysis during the cipher design phase, and shows that these strong structural distinguishers could have been avoided during this phase.

Keywords: ARADI· Algebraic attacks · Division property · Cube attacks · Toffoli gates

1 Introduction

In the evolving cryptography landscape, block ciphers play a pivotal role in safeguarding sensitive data, particularly in memory encryption systems where low latency is crucial. Memory encryption is vital for protecting data as it moves

between the processor and memory, ensuring confidentiality and integrity against attacks targeting vulnerable memory contents.

Memory encryption faces several challenges, primarily balancing performance and security. Achieving low latency is critical to prevent delays and excessive energy consumption, especially in real-time and mobile applications and embedded systems. Additionally, memory encryption must resist physical attacks. Designing secure ciphers that can withstand these challenges, particularly in multicore and virtualized environments, while protecting non-volatile memory from attacks remains a complex challenge.

Low-latency block ciphers have been a topic of significant interest, especially for memory encryption systems. Several designs have emerged over the years, focusing on achieving high performance without compromising cryptographic strength. Among these, PRINCE [12] stands as an early influential design, followed by improvements such as PRINCEv2 [13]. Other noteworthy examples include MANTIS [7], which targets lightweight applications, and QARMA [3], designed with flexibility and security in mind. Recent designs continue to push the boundaries of efficiency and security, including SPEEDY [25], ORTHROS [6], QARMAv2 [4]. SCARF [14], BipBip [8], Sonic and SuperSonic [9], Gleook [1] and Twinkle [29].

ARADI, a modern low-latency block cipher with 128-bit block size and 256-bit key size, mainly designed for memory encryption use-case, was recently made public by the US National Security Agency (NSA) [18]. Unlike other academic publications of new symmetric primitives, the choices behind the design of ARADI were not made explicit by the authors. So, this primitive's resilience must be thoroughly investigated against a range of cryptanalytic techniques to have confidence in its security.

This article delves into the algebraic cryptanalysis of ARADI, leveraging algebraic methods to explore potential weaknesses in its structure. In the following, we first discuss the related works on ARADI, and then describe our contributions in detail.

1.1 Related Works on ARADI

After its publication not much research on the security of ARADI has been published after its publication, and nothing is peer-reviewed yet. In Bellini et al. [10], the authors utilize the automated tool CLAASP [11] to analyze the ARADI cipher using a combination of techniques, including: statistical black-box analysis such as avalanche and diffusion tests, (up to 9 rounds) differential and linear trails, (8 rounds) impossible trails, and (5-round) neural distinguishers. They also include a preliminary algebraic analysis revealing an integral distinguisher reaching 7 rounds of ARADI with 2^{124} data. The authors also show that in some instances, the algebraic degree grows more slowly, reaching only 60 after 5 rounds, which leads to an integral distinguisher with a data complexity of 2^{61}.

In another unpublished note, Avanzi, Dunkelman, and Ghosh [5] evaluate ARADI's recent design and question whether it offers significant improvements over existing low-latency block ciphers in terms of area-latency trade-offs. They

highlight an issue related to the LLAMA mode of operation, particularly with the varying length of initialization vectors (IVs), that compromises the integrity and confidentiality of ciphertexts. Later, the same weakness was acknowledged by ARADI's designers in their updated Eprint report.

1.2 Our Contributions

The work by Bellini et al. [10] did not consider any structural aspects of ARADI. Instead, the provided results are obtained by modeling ARADI in CLAASP, and then running the toolkit. Avanzi et al.'s [5] work also did not look at any cipher-specific property. On the other hand, this work takes an in-depth look inside the structure of ARADI and reveals new structural weaknesses leading to several algebraic distinguishers. Our contributions in detail are as follows.

Degree bounds with division property. We provide the mixed integer linear programming-based modeling of the three-subset division property of ARADI (see Section 3). Consequently, using this model, we obtain better algebraic distinguishers than Bellini et al. in terms of number of rounds and data complexities.

New structural property of ARADI. We present a structural property called *weakly-composed-Toffoli gates* that exploits the composition of Toffoli gates in the nonlinear layer. In simple terms, the state update with the first two Toffoli gates strongly depends on the algebraic normal form of the output bits of two state words. We introduce a general theoretical framework that describes this property (Theorem 1). The notable feature of the property is that it permits the extension of an r-round algebraic distinguisher to $r+1$ rounds with the same data complexity as of r-round distinguisher. The $r+1$ rounds distinguisher, though depending on the key bits, always has equal cube-sum (4) values in two of the state words. It is to be noted that such a property has never been found before in any state-of-the-art block cipher. Also, the dependence on key bits makes it difficult to obtain them directly by division property.

Improved distinguishers. Using the weakly-composed-Toffoli gates property in Theorem 1, we give several algebraic distinguishers reaching 8 rounds of ARADI. Compared to Bellini et al. and division property-based distinguishers, the data complexities of these structural distinguishers are lower. Table 1 compares all distinguishers in three different settings.

(Im)possibility of fixing the structural distinguishers. We show that the aforementioned property is independent of the choice of the linear layer. Furthermore, we investigate all possible combinations of Toffoli gates for 4 state words and find that results similar to Theorem 1 still hold. Consequently, we show that it is impossible to prevent this property without re-designing the nonlinear layer.

Table 1. A comparison of algebraic distinguishers for ARADI with Theorem 1 (using cubes from Table 4), three-subset division property (using cubes from Table 2) and Bellini et al. [10].

Round	Data ($\log_2(\cdot)$ scale)		
	Theorem 1	Division property	Bellini et al. [10]
4	4	4	28
5	16	16	84
6	84	92	113
7	113	116	124
8	124	124	-

Key recovery attack. We report a key recovery attack on 10 rounds ARADI using the 8-round distinguisher. The attack requires 2^{124} data and 2^{177} time. Though the key recovery attack is not the focus of this work and provided as a side result, it is the first one in literature using the algebraic distinguishers.

We provide the source codes of our implementations for verification and computation of values provided in all tables at https://github.com/Crypto-TII/aradi_structural_algebraic_distinguisher.

1.3 Outline of the Paper

The remainder of the paper is organized as follows. In Section 2, we recall the basics of algebraic attacks and give the specification of ARADI along with our attack setting. Section 3 provides the MILP modeling for the three-subset division trails of ARADI and our experimental results on algebraic degrees. Sections 4 and 5 introduces the weakly-composed-Toffoli gates structural property of ARADI and the related distinguishers, respectively. In Section 6, we investigate the possibility of fixing these structural distinguishers. We provide a key recovery attack on 10 rounds ARADI in Section 7. Finally, we conclude the paper in Section 8 with future research directions.

2 Preliminaries

In this section, we describe the notation used throughout the paper, briefly explain the fundamentals of algebraic attacks, and give an overview of the ARADI block cipher. In the end, we discuss the attack setting for ARADI that we use throughout the paper.

2.1 Notation

Let $\mathcal{A}$ be a set and $\tilde{\mathcal{A}}$ be the complementary set of $\mathcal{A}$. We denote its cardinality by $|\mathcal{A}|$. Given two sets A and B, $A \cup B$ denotes the union of A and B. By $\mathbb{F}_2 = \{0, 1\}$, we denote the finite field with two elements and $\mathbb{F}_2^n$ denotes the

n-dimensional vector space of $\mathbb{F}_2$. Let $x, y \in \mathbb{F}_2^n$ such that $x = (x_0, \ldots, x_{n-1})$ and $y = (y_0, \ldots, y_{n-1})$. Then, we use $x \odot y$ and $x + y$ to denote the bitwise AND and bitwise XOR, respectively. Note that '$+$' may also denote other kinds of addition (such as integer addition or field addition), and the meaning should be clear from the context. We say $x \preceq y$, if $x_i \leq y_i$ for all i.

Let $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ be a Boolean function. We define its Algebraic Normal Form (ANF) as $f(x) = \sum_{u \in \mathbb{F}_2^n} a_u x^u$ where $a_u \in \mathbb{F}_2$ and $x^u = \prod_{i=0}^{n-1} x_i^{u_i}$ is a monomial. We denote the coefficient a_u of the monomial x^u in the ANF of f by $\mathsf{Coe}_f(x^u)$ and therefore, $\mathsf{Coe}_f(x^u) = a_{\boldsymbol{u}}$. For a given u, the coefficient $\mathsf{Coe}_f(x^u)$ can be computed as follows [15].

$$\mathsf{Coe}_f(x^u) = \sum_{x \preceq u} f(x)\,. \tag{1}$$

The algebraic degree of f is given as

$$\deg(f) = \max_{u \in \mathbb{F}_2^n} \{\boldsymbol{wt}(u) \mid \mathsf{Coe}_f(x^u) \neq 0\}$$

where $\boldsymbol{wt}(u)$ represents the Hamming weight of the vector u.

2.2 Algebraic Attacks

In our context, algebraic attacks are common cryptanalytic techniques against symmetric key ciphers. The main idea of an algebraic attack is to exploit the algebraic degree of a Boolean function, i.e., for an n-variable Boolean function with $\deg(f) = d$ (with $d < n - 1$), summing the outputs of f over any $(d+1)$-dimensional affine subspace, the final value will always be zero. This idea is at the core of several variants of algebraic attacks, namely higher-order differential distinguishers [24], integral distinguishers [16,23], cube attacks [17,2].

Consider a block cipher with n-bit block size and m-bit key size. Let $p = (p_0, \ldots, p_{n-1})$ be n public (plaintext) bit variables and $k = (k_0, \ldots, k_{m-1})$ be m secret bit variables. Then, any output bit of a block cipher can be regarded as a Boolean function $f : \mathbb{F}_2^m \times \mathbb{F}_2^n \mapsto \mathbb{F}_2$ given by

$$f(k, p) = \sum_{u \in \mathbb{F}_2^n} \sum_{v \in \mathbb{F}_2^m} \mathsf{Coe}_f(p^u \cdot k^v) \cdot p^u \cdot k^v \tag{2}$$

Given a set of indices $I = \{i_0, i_1, \ldots, i_{d-1}\} \subset \{0, 1, \ldots, n-1\}$ with $p(I)$ denoting the monomial $\prod_{i \in I} p_i$, (2) can be alternatively written [17] as

$$f(k, p) = p(I) \cdot t(\underset{i \in \tilde{I}}{p_i}; k_0, \ldots, k_{m-1}) + q(p_0, \ldots, p_{n-1}, k_0, \ldots, k_{m-1}), \tag{3}$$

where $\mathsf{Coe}_f(p(I)) = t(p_{i_{i \in \tilde{I}}}; k_0, \ldots, k_{m-1})$ and the Boolean function q misses at least one variable from $\{p_i \mid i \in I\}$. Let $\mathcal{C}_I$ denote the set of n-bit vectors p such that $\{p_i\}_{i \in I}$ can take all possible values while $\{p_i\}_{i \in \tilde{I}}$ are fixed to some

constant. We call $\mathcal{C}_I$ as the *d-dimensional cube* corresponding to *cube indices* I. Furthermore, for a fixed key, we have

$$\sum_{p \in \mathcal{C}_I} f(p) = \mathsf{Coe}_f(p(I)) \,. \tag{4}$$

The left term in (4) is called as *cube-sum*. If the algebraic degree in *cube variables* $\{p_i \mid i \in I\}$ is less than the cube-dimension d for all keys, then the cube-sum is always 0, i.e., $\mathsf{Coe}_f(p(I)) = 0$.

A useful tool to check whether the value of $\mathsf{Coe}_f(p(I))$ equals 0 is the division property, introduced by Todo [26]. The technique offers a systematic and automated approach to probe the algebraic structure of Boolean functions, especially when their ANF is too complex to compute [28,20,31,27,19,30]. Specifically, the bit-based division property can give an indication of the value of $\mathsf{Coe}_f(p(I))$ in the targeted Boolean function [20,27,19,30,21,22]. We introduce the technical details of division property on-site in Section 3 when immediately necessary.

2.3 Specification of ARADI

ARADI is a low-latency block cipher proposed by the US National Security Agency researchers Greene et al. [18]. The block size and key size are 128 and 256 bits, respectively. The ARADI round function is an substitution permutation network design and consists of three operations, namely the S-box layer π, the i-th linear map Λ_i, and the i-th round key addition τ_{k_i}. The cipher consists of 16 rounds and is given by

$$\tau_{k_{16}} \circ \bigcirc_{i=0}^{15} (\Lambda_{i \bmod 4} \circ \pi \circ \tau_{k_i}), \tag{5}$$

where the composition is read from right to left.

The round function operates on a 128-bit state arranged into four 32-bit words $\mathsf{W}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}$. We now briefly explain the individual operations of round function.

The S-box layer π. The S-box layer is based on the Toffoli gate applied to three 32-bit words a, b, c, so that $(a, b, c) \mapsto (a, b, c + a \odot b)$. The input state $(\mathsf{W}, \mathsf{X}, \mathsf{Y}, \mathsf{Z})$ is transformed by π as follows:

$$\mathsf{X} \leftarrow \mathsf{X} + \mathsf{W} \odot \mathsf{Y},\ \mathsf{Z} \leftarrow \mathsf{Z} + \mathsf{X} \odot \mathsf{Y},\ \mathsf{Y} \leftarrow \mathsf{Y} + \mathsf{W} \odot \mathsf{Z},\ \mathsf{W} \leftarrow \mathsf{W} + \mathsf{X} \odot \mathsf{Z}. \tag{6}$$

The linear layer Λ_i. At round i, the input state $(\mathsf{W}, \mathsf{X}, \mathsf{Y}, \mathsf{Z})$ is transformed by Λ_i as follows:

$$\Lambda_i(\mathsf{W}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}) \rightarrow (L_i(\mathsf{W}), L_i(\mathsf{X}), L_i(\mathsf{Y}), L_i(\mathsf{Z})), \tag{7}$$

where L_i is an involutory linear map on 32-bit words. Let the 32-bit input to L_i be composed of two 16-bit words u and l. Then L_i is given by

$$(u, l) \rightarrow (u + S_{16}^{a_i}(u) + S_{16}^{c_i}(l),\ l + S_{16}^{a_i}(l) + S_{16}^{b_i}(u)), \tag{8}$$

where the operation $S_{16}^{m}(\cdot)$ denotes the left circular shift of a 16-bit word by m positions. We refer the reader to [18][Section 3.1] for the exact shift offsets.

The key addition layer. This operation XORs a 128-bit round key to the state. The round keys are generated by the dedicated key scheduling algorithm as explained below.

The key schedule algorithm. ARADI's key schedule operates on an array of eight 32-bit registers. Let $K_0^i, K_1^i, \ldots, K_7^i$ denote the state at i-th step, then the i-th 128-bit round key rk^i equals $K_0^i \| K_1^i \| K_2^i \| K_3^i$ for even index round and $K_4^i \| K_5^i \| K_6^i \| K_7^i$ otherwise.

At each step, words $\{0, 1\}$ and $\{4, 5\}$ (resp. $\{2, 3\}$ and $\{6, 7\}$) of state are updated by the 64-bit invertible linear map M_0 (resp. M_1). Then, a word permutation P_j is applied where $P_0 = (12)(56)$ and $P_1 = (14)(36)$, and j is the round modulo 2.

The linear maps M_0 and M_1 operates on a pair of 32-bit words a and b are given by

$$\begin{aligned} M_0(a, b) &= (S_{32}^1(a) + b, S_{32}^3(b) + S_{32}^1(a) + b) \\ M_1(a, b) &= (S_{32}^9(a) + b, S_{32}^{28}(b) + S_{32}^9(a) + b) \end{aligned} \tag{9}$$

where S_{32} is a left circular shift on a 32-bit word.

2.4 Attack Setting for ARADI

In our analysis of ARADI, we denote the cipher's state after r rounds using the 32-bit words $\mathsf{W}^r, \mathsf{X}^r, \mathsf{Y}^r, \mathsf{Z}^r$. The Boolean functions of the i-th bit of each word after r rounds is represented by $\mathsf{W}_i^r, \mathsf{X}_i^r, \mathsf{Y}_i^r, \mathsf{Z}_i^r$. The sets of bit indices for the input words (plaintext) $\mathsf{W}^0, \mathsf{X}^0, \mathsf{Y}^0, \mathsf{Z}^0$ are denoted as $\mathcal{I}_\mathsf{w}, \mathcal{I}_\mathsf{x}, \mathcal{I}_\mathsf{y}, \mathcal{I}_\mathsf{z} \subseteq \{0, 1, \ldots, 31\}$.

We refer the indices in $\mathcal{I}_\mathsf{w}, \mathcal{I}_\mathsf{x}, \mathcal{I}_\mathsf{y}, \mathcal{I}_\mathsf{z}$ collectively as the cube indices of dimension d, and satisfying $|\mathcal{I}_\mathsf{w}| + |\mathcal{I}_\mathsf{x}| + |\mathcal{I}_\mathsf{y}| + |\mathcal{I}_\mathsf{z}| = d < 127$. These index sets $\mathcal{I}_\mathsf{w}, \mathcal{I}_\mathsf{x}, \mathcal{I}_\mathsf{y}, \mathcal{I}_\mathsf{z}$ are sometimes referred to as cube-index sets. The corresponding set of plaintext variables

$$\{\mathsf{W}_i^0 \mid i \in \mathcal{I}_\mathsf{w}\} \bigcup \{\mathsf{X}_i^0 \mid i \in \mathcal{I}_\mathsf{x}\} \bigcup \{\mathsf{Y}_i^0 \mid i \in \mathcal{I}_\mathsf{y}\} \bigcup \{\mathsf{Z}_i^0 \mid i \in \mathcal{I}_\mathsf{z}\} \tag{10}$$

is referred to as cube variables.

A monomial $\boldsymbol{m}$ corresponding to cube-index sets $\mathcal{I}_{\mathsf{w}}, \mathcal{I}_{\mathsf{x}}, \mathcal{I}_{\mathsf{y}}, \mathcal{I}_{\mathsf{z}}$ is defined as follows.

$$\begin{aligned}\boldsymbol{m} &\coloneqq \mathsf{W}^0(\mathcal{I}_{\mathsf{w}})\mathsf{X}^0(\mathcal{I}_{\mathsf{x}})\mathsf{Y}^0(\mathcal{I}_{\mathsf{y}})\mathsf{Z}^0(\mathcal{I}_{\mathsf{z}}) \\ &= \prod_{i \in \mathcal{I}_{\mathsf{w}}} \mathsf{W}_i^0 \cdot \prod_{i \in \mathcal{I}_{\mathsf{x}}} \mathsf{X}_i^0 \cdot \prod_{i \in \mathcal{I}_{\mathsf{y}}} \mathsf{Y}_i^0 \cdot \prod_{i \in \mathcal{I}_{\mathsf{z}}} \mathsf{Z}_i^0 \end{aligned} \tag{11}$$

In the remaining part of the paper, we aim to find $\mathcal{I}_{\mathsf{w}}, \mathcal{I}_{\mathsf{x}}, \mathcal{I}_{\mathsf{y}}, \mathcal{I}_{\mathsf{z}}$ so that $\mathsf{Coe}_{\mathsf{W}_i^r}(\boldsymbol{m})$, $\mathsf{Coe}_{\mathsf{X}_i^r}(\boldsymbol{m})$, $\mathsf{Coe}_{\mathsf{Y}_i^r}(\boldsymbol{m})$ and $\mathsf{Coe}_{\mathsf{Z}_i^r}(\boldsymbol{m})$ are either zero (degree in cube variables set (10)) or have certain structural relationships.

3 Degree Bounds with Division Property

In this section, we present integral distinguishers for the round-reduced ARADI by computing the upper bounds on the algebraic degree using the three-subset bit-based division property (3SBDP) [20]. We begin with a brief description of Mixed Integer Linear Programming (MILP) models for the propagation rules of the 3SBDP in ARADI. Subsequently, we provide degree bounds for round-reduced ARADI, leading to several integral distinguishers.

3.1 MILP Modeling

ARADI block cipher consists of the following basic operations: bitwise XOR, bitwise AND, bitwise ROTATION, and bitwise XOR with a constant. To model these operations in MILP, along with the bitwise COPY, which is essential for the division property search, we need the following constraints. Let $a, b, b_1, b_2, \ldots, b_n$ be binary variables. Then, the above operations can be modeled as follows [20].

- **Bitwise XOR:** $(b_1, \ldots, b_n) \xrightarrow{\text{XOR}} a : a = b_1 + b_2 + \ldots + b_n$
- **Bitwise AND:** $(b_1, \ldots, b_n) \xrightarrow{\text{AND}} a : a = b_i$ for $1 \leq i \leq n$
- **Bitwise COPY:** $a \xrightarrow{\text{COPY}} (b_1, \ldots, b_n) : a \geq b_i$ for $1 \leq i \leq n$, and $b_1 + b_2 + \ldots + b_n \geq x$
- **Bitwise XOR with constant 1:** $a \xrightarrow{\text{XOR+1}} b : b \geq a$.

Algorithm 1 provides the three-subset bit-based division property MILP modeling of r rounds ARADI using the above basic operations. The exact modeling of underlying functions RoundkeyXOR, S-box, Linear-map in Algorithm 1 is discussed in Appendix A.

3.2 Upper Bounds on Algebraic Degree

We use Algorithm 1 to compute the upper bounds of algebraic degree of a monomial corresponding to the cube-index sets $\mathcal{I}_{\mathsf{w}}, \mathcal{I}_{\mathsf{x}}, \mathcal{I}_{\mathsf{y}}$ and $\mathcal{I}_{\mathsf{z}}$ in the ANF of W_i^r, X_i^r, Y_i^r, and Z_i^r for a given output bit i and after r rounds. The entire

Algorithm 1: MILP model for 3SBDP of r rounds ARADI

Input: Empty model $\mathcal{M}$, number of rounds r, key variables $k_0, \ldots, k_{255}$
Output: MILP model $\mathcal{M}$

$\mathcal{M}.addVar \leftarrow w_i^0, x_i^0, y_i^0, z_i^0$, for $i = 0, \ldots, 31$
$\mathcal{M}.addVar \leftarrow rk_i^0, a_i^0$, for $i = 0, \ldots, 127$
RoundkeyXOR$(\mathcal{M}, a_0^0, \ldots, a_{31}^0, w_0^0, \ldots, w_{31}^0, rk_0^0, \ldots, rk_{31}^0)$
RoundkeyXOR$(\mathcal{M}, a_{32}^0, \ldots, a_{63}^0, x_0^0, \ldots, x_{31}^0, rk_{32}^0, \ldots, rk_{63}^0)$
RoundkeyXOR$(\mathcal{M}, a_{64}^0, \ldots, a_{95}^0, y_0^0, \ldots, y_{31}^0, rk_{64}^0, \ldots, rk_{95}^0)$
RoundkeyXOR$(\mathcal{M}, a_{96}^0, \ldots, a_{127}^0, z_0^0, \ldots, z_{31}^0, rk_{96}^0, \ldots, rk_{127}^0)$
for $j = 0$ *to* $r - 1$ **do**
  $\mathcal{M}.addVar \leftarrow b_i^j, c_i^j, d_i^j, e_i^j$, for $i = 0, \ldots, 31$
  S-box$(\mathcal{M}, b_0^j, \ldots, b_{31}^j, c_0^j, \ldots, c_{31}^j, d_0^j, \ldots, d_{31}^j, e_0^j, \ldots, e_{31}^j, a_0^j, \ldots, a_{127}^j)$
  $\mathcal{M}.addVar \leftarrow p_i^j, q_i^j, s_i^j, t_i^{j+1}$, for $i = 0, \ldots, 31$
  Linear-map$(\mathcal{M}, j \bmod 4, p_0^j, \ldots, p_{31}^j, b_0^j, \ldots, b_{31}^j)$
  Linear-map$(\mathcal{M}, j \bmod 4, q_0^j, \ldots, q_{31}^j, c_0^j, \ldots, c_{31}^j)$
  Linear-map$(\mathcal{M}, j \bmod 4, s_0^j, \ldots, s_{31}^j, d_0^j, \ldots, d_{31}^j)$
  Linear-map$(\mathcal{M}, j \bmod 4, t_0^j, \ldots, t_{31}^j, e_0^j, \ldots, e_{31}^j)$
  $\mathcal{M}.addVar \leftarrow w_i^{j+1}, x_i^{j+1}, y_i^{j+1}, z_i^{j+1}$, for $i = 0, \ldots, 31$
  $\mathcal{M}.addVar \leftarrow rk_i^{j+1}$, for $i = 0, \ldots, 127$
  RoundkeyXOR$(\mathcal{M}, w_0^{j+1}, \ldots, w_{31}^{j+1}, p_0^j, \ldots, p_{31}^j, rk_0^j, \ldots, rk_{31}^j)$
  RoundkeyXOR$(\mathcal{M}, x_0^{j+1}, \ldots, x_{31}^{j+1}, q_0^j, \ldots, q_{31}^j, rk_{32}^j, \ldots, rk_{63}^j)$
  RoundkeyXOR$(\mathcal{M}, y_0^{j+1}, \ldots, y_{31}^{j+1}, s_0^j, \ldots, s_{31}^j, rk_{64}^j, \ldots, rk_{95}^j)$
  RoundkeyXOR$(\mathcal{M}, z_0^{j+1}, \ldots, z_{31}^{j+1}, t_0^j, \ldots, t_{31}^j, rk_{96}^j, \ldots, rk_{127}^j)$
return MILP model $\mathcal{M}$

Algorithm 2: MILP model for computing the upper bound on degree

Input: Empty model $\mathcal{M}$, number of rounds r, key variables $k_0, \ldots, k_{255}$, bit position *target*, Indices sets $\mathcal{I}_{\mathsf{W}}$, $\mathcal{I}_{\mathsf{X}}$, $\mathcal{I}_{\mathsf{Y}}$, $\mathcal{I}_{\mathsf{Z}}$
Output: Degree upper bound of targeted output bit

Model R rounds ARADI using Algorithm 1
$S = w_0^r \| \ldots \| w_{31}^r \| x_0^r \| \ldots \| x_{31}^r \| y_0^r \| \ldots \| y_{31}^r \| z_0^r \| \ldots \| z_{31}^r$
for $i = 0$ *to* 127 **do**
  if $i = target$ **then**
    $\mathcal{M}.addConstr(S_i = 1)$
  else
    $\mathcal{M}.addConstr(S_i = 0)$
  $\mathcal{M}.setObjective(\sum_{i \in \mathcal{I}_{\mathsf{W}}} w_i^0 + \sum_{i \in \mathcal{I}_{\mathsf{X}}} x_i^0 + \sum_{i \in \mathcal{I}_{\mathsf{Y}}} y_i^0 + \sum_{i \in \mathcal{I}_{\mathsf{Z}}} z_i^0, \mathsf{Maximize})$
return Objective value

algorithm to compute the degree upper bounds for each state bit is explained in Algorithm 2.

In Table 2, we give the degree bounds for the monomials corresponding to the cube-index sets in the second column, for up to 8 rounds using Algorithm

2. We provide the minimum and maximum degrees computed for W_i^r, X_i^r, Y_i^r and Z_i^r for $i \in \{0, 1, \ldots, 31\}$. The degrees of W_i^r and Y_i^r remain the same for the respective bits, and similarly, the degrees of X_i^r and Z_i^r remain the same. Therefore, we present the degree bounds for W_i^r and Y_i^r in a single column, and the degree bounds for X_i^r and Z_i^r in another column.

For rounds 4 and 5, we find several indices $i \in \{0, 1, \ldots, 31\}$ for which the degrees remain `min` (column 3 and 5). In rounds 6, 7, and 8 the degrees remain constant and lower than the dimension of the cubes for all i.

Table 2. Index sets and the corresponding minimum and maximum algebraic degrees of W_i^r, X_i^r, Y_i^r and Z_i^r in all $i \in \{0, 1, \ldots, 31\}$ up to 8 rounds using division property.

Round r	Indices	Degree in $\mathsf{W}_i^r, \mathsf{Y}_i^r$		Degree in $\mathsf{X}_i^r, \mathsf{Z}_i^r$		Cube dimension
		`min`	`max`	`min`	`max`	
4	$\mathcal{I}_\mathsf{W} = \{0\}$ $\mathcal{I}_\mathsf{X} = \{0\}$ $\mathcal{I}_\mathsf{Y} = \{0\}$ $\mathcal{I}_\mathsf{Z} = \{0\}$	3	4	3	4	4
5	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 4\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 4\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 4\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 4\}$	15	16	15	16	16
6	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 22\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 22\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 22\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 22\}$	92	92	90	90	92
7	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 28\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 28\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 28\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 28\}$	116	116	115	115	116
8	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 30\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 30\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 30\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 30\}$	124	124	123	123	124

Remark 1. We use Gurobi 11 as the underlying MILP solver and run all experiments in this paper on a machine with two AMD EPYC processors (x86_64 architecture).

In the next section, we will explain a structural weakness in the non-linear layer of the cipher, which will lead to better distinguishers.

4 New Structural Property of ARADI

In this section, we present a structural property of ARADI's round function by exploiting the composition of Toffoli gates. We call it the *weakly-composed-Toffoli* gates property. This property allows to extend a given r rounds integral distinguisher to $r+1$ rounds integral distinguisher without changing the data complexity.

We first explain this property with an example and then provide its mathematical details in Theorem 1. Recall that we use $\mathsf{W}_i^r, \mathsf{X}_i^r, \mathsf{Y}_i^r$, and Z_i^r to represent the Boolean functions corresponding to the i-th bit of the 32-bit words $\mathsf{W}^r, \mathsf{X}^r, \mathsf{Y}^r$, and Z^r of ARADI after r rounds, respectively. Furthermore, $\mathcal{I}_\mathsf{w}$, $\mathcal{I}_\mathsf{x}$, $\mathcal{I}_\mathsf{y}, \mathcal{I}_\mathsf{z} \subseteq \{0, 1, \ldots, 31\}$ denote the sets corresponding to bit indices of input words (plaintext words) $\mathsf{W}^0, \mathsf{X}^0, \mathsf{Y}^0$, Z^0 (see Section 2.4).

4.1 Weakly-composed-Toffoli Gates

We start with an example to explain what we mean by weakly-composed-Toffoli Gates for ARADI.

Example 1. Let $\mathcal{I}_\mathsf{w} = \{0, 1, \ldots, 15\}$, $\mathcal{I}_\mathsf{x} = \phi$, $\mathcal{I}_\mathsf{y} = \phi$ and $\mathcal{I}_\mathsf{z} = \phi$. We set all plaintext bits corresponding to $\tilde{\mathcal{I}}_\mathsf{w}$, $\tilde{\mathcal{I}}_\mathsf{x}$, $\tilde{\mathcal{I}}_\mathsf{y}$, $\tilde{\mathcal{I}}_\mathsf{z}$ as constant zero. Then, after 4 rounds, the cube-sum of the outputs of 2^{16} plaintexts is always zero. More precisely for the monomial $\boldsymbol{m} = \mathsf{W}^0(\mathcal{I}_\mathsf{w})\mathsf{X}^0(\mathcal{I}_\mathsf{x})\mathsf{Y}^0(\mathcal{I}_\mathsf{y})\mathsf{Z}^0(\mathcal{I}_\mathsf{z})$, we have $\mathsf{Coe}_{\mathsf{W}_i^4}(\boldsymbol{m}) = 0$, $\mathsf{Coe}_{\mathsf{X}_i^4}(\boldsymbol{m}) = 0$, $\mathsf{Coe}_{\mathsf{Y}_i^4}(\boldsymbol{m}) = 0$ and $\mathsf{Coe}_{\mathsf{Z}_i^4}(\boldsymbol{m}) = 0$, for all $i = 0, \ldots, 31$. Notice that this can be easily verified with the division property (Section 3).

Interestingly, we also find that the cube-sum of the outputs of 2^{16} plaintexts after 5 rounds are exactly equal in two of the state words. In particular, we find that

$$\mathsf{Coe}_{\mathsf{X}_i^5}(\boldsymbol{m}) = \mathsf{Coe}_{\mathsf{Z}_i^5}(\boldsymbol{m}), \text{ for all } i = 0, \ldots, 31. \tag{12}$$

We repeated Example 1 with 2^{16} keys and for all these keys, we observe this structural pattern. This can be verified using ARADI encryption source code provided at https://github.com/Crypto-TII/aradi_structural_algebraic_distinguisher. Here, we give the explicit values of these coefficients for some keys in Table 3.

In order to understand the reasoning behind the structural pattern in Table 3, we look at the composition of Toffoli gates and how the state words are updated in the 5-th round. The first Toffoli gate is applied on $(\mathsf{W}^4, \mathsf{Y}^4, \mathsf{X}^4)$ resulting in

$$\mathsf{X}^5 = L_0(\mathsf{X}^4 + \mathsf{W}^4 \odot \mathsf{Y}^4) = L_0(\mathsf{X}^4) + L_0(\mathsf{W}^4 \odot \mathsf{Y}^4), \tag{13}$$

where L_0 is the linear map.

Next, the second Toffoli gate is applied on $(\mathsf{X}^4 + \mathsf{W}^4 \odot \mathsf{Y}^4, \mathsf{Y}^4, \mathsf{Z}^4)$ which gives the value of Z^5 as follows.

$$\begin{aligned} \mathsf{Z}^5 &= L_0(\mathsf{Z}^4 + (\mathsf{X}^4 + \mathsf{W}^4 \odot \mathsf{Y}^4) \odot \mathsf{Y}^4) \\ &= L_0(\mathsf{Z}^4) + L_0(\mathsf{X}^4 \odot \mathsf{Y}^4) + L_0(\mathsf{W}^4 \odot \mathsf{Y}^4) \end{aligned} \tag{14}$$

Table 3. A 5-round distinguisher of ARADI with equal coefficients of monomial $\boldsymbol{m}$ (marked in red) in two words after 5 rounds. Here $\mathcal{I}_{\mathsf{w}} = \{0, 1, \ldots, 15\}$, $\mathcal{I}_{\mathsf{x}} = \phi$, $\mathcal{I}_{\mathsf{y}} = \phi$ and $\mathcal{I}_{\mathsf{z}} = \phi$. The plaintext bits corresponding to $\tilde{\mathcal{I}}_{\mathsf{w}}$, $\tilde{\mathcal{I}}_{\mathsf{x}}$, $\tilde{\mathcal{I}}_{\mathsf{y}}$, $\tilde{\mathcal{I}}_{\mathsf{z}}$ are set as constant zero.

Master key	Round r	Coefficients of monomial $\boldsymbol{m}$ in word W^r	X^r	Y^r	Z^r
a77a0507 6181d046 dc8b0dd8 8e1c60da	4	0	0	0	0
b5972211 8b06df2a c5825749 b61309a3	5	7dad17a7	17171c1c	3ec42fe5	17171c1c
4c11e8d1 f5abb1a2 e03d40be cd6d268c	4	0	0	0	0
b882ef59 64db2c6f 565612b2 3b6a5164	5	47fb9d78	4949e8e8	77957aba	4949e8e8
62a2ddfa c1dda4c7 8c5b112 322d62c3	4	0	0	0	0
a82ba564 167d9e64 639aaa1d ae052fb0	5	5ce9b27a	41417a7a	787f1a40	41417a7a
5ac786d b0780d6 16bae661 6c421677	4	0	0	0	0
279cb2df 42e60421 5cb71c1f 10dcdea4	5	f624a230	11112d2d	20725e2b	11112d2d

In (13) and (14), we observe that the common term is $L_0(\mathsf{W}^4 \odot \mathsf{Y}^4)$. The cube-sums of $L_0(\mathsf{X}^4)$ and $L_0(\mathsf{Z}^4)$ are zero because the cube-sums of X^4 and Z^4 are zero, and the linear layer do not increase the algebraic degree. So, the only way the cube-sum of X^5 and Z^5 can be equal is when the cube-sum of $L_0(\mathsf{X}^4 \odot \mathsf{Y}^4)$ is zero. This can happen when the degree of $\mathsf{X}^4 \odot \mathsf{Y}^4$ in cube variables from $\mathcal{I}_{\mathsf{w}}$ is at most 15, i.e., the monomial $\boldsymbol{m}$ is missing in the ANF of each bit of $\mathsf{X}^4 \odot \mathsf{Y}^4$. Once this is satisfied, the same $L_0(\mathsf{W}^4 \odot \mathsf{Y}^4)$ contributes in the cube-sum values after 5 rounds. Hence, the cube-sum values of X^5 and Z^5 are equal in Table 3.

The above distinguisher reveals a potential weakness of the composition of Toffoli gates in ARADI's nonlinear layer. In the following section, we give a theoretical description of this weakness and discuss how to exploit it to construct an $r+1$-round distinguisher from a r-round distinguisher.

4.2 Mathematical Description

Theorem 1. *Let $r \geq 2$ and $\mathcal{I}_{\mathsf{w}}, \mathcal{I}_{\mathsf{x}}, \mathcal{I}_{\mathsf{y}}, \mathcal{I}_{\mathsf{z}} \subseteq \{0, 1, \ldots, 31\}$ such that $|\mathcal{I}_{\mathsf{w}}| + |\mathcal{I}_{\mathsf{x}}| + |\mathcal{I}_{\mathsf{y}}| + |\mathcal{I}_{\mathsf{z}}| \leq 127$. Define the momomial*

$$\boldsymbol{m} = \prod_{i \,\in\, \mathcal{I}_{\mathsf{w}}} \mathsf{W}_i^0 \cdot \prod_{i \,\in\, \mathcal{I}_{\mathsf{x}}} \mathsf{X}_i^0 \cdot \prod_{i \,\in\, \mathcal{I}_{\mathsf{y}}} \mathsf{Y}_i^0 \cdot \prod_{i \,\in\, \mathcal{I}_{\mathsf{z}}} \mathsf{Z}_i^0 \,.$$

For each $i \in \{0, \ldots, 31\}$, suppose the following conditions are satisfied.

(a) $Coe_{\mathsf{X}_i^r}(\boldsymbol{m}) = 0$
(b) $Coe_{\mathsf{Z}_i^r}(\boldsymbol{m}) = 0$
(c) $Coe_{\mathsf{X}_i^r \odot \mathsf{Y}_i^r}(\boldsymbol{m}) = 0$

Then, the following holds:

$$Coe_{\mathsf{X}_i^{r+1}}(\boldsymbol{m}) = Coe_{\mathsf{Z}_i^{r+1}}(\boldsymbol{m})$$

Proof. Consider the r-th round of ARADI. The application of the first Toffoli gate maps $(\mathsf{W}^r, \mathsf{Y}^r, \mathsf{X}^r) \mapsto (\mathsf{W}^r, \mathsf{Y}^r, \mathsf{W}^r \odot \mathsf{Y}^r + \mathsf{X}^r)$. So, for each $i \in \{0, \ldots, 31\}$, we have

$$\mathsf{Coe}_{\mathsf{W}_i^r \odot \mathsf{Y}_i^r + \mathsf{X}_i^r}(\boldsymbol{m}) = \mathsf{Coe}_{\mathsf{W}_i^r \odot \mathsf{Y}_i^r}(\boldsymbol{m}) + \mathsf{Coe}_{\mathsf{X}_i^r}(\boldsymbol{m}) \tag{15}$$

Since $\mathsf{Coe}_{\mathsf{X}_i^r}(\boldsymbol{m}) = 0$ (by condition (a)), (15) reduces to

$$\mathsf{Coe}_{\mathsf{W}_i^r \odot \mathsf{Y}_i^r + \mathsf{X}_i^r}(\boldsymbol{m}) = \mathsf{Coe}_{\mathsf{W}_i^r \odot \mathsf{Y}_i^r}(\boldsymbol{m}). \tag{16}$$

Similarly, the second Toffoli gate maps

$$(\mathsf{X}^r + \mathsf{W}^r \odot \mathsf{Y}^r, \mathsf{Y}^r, \mathsf{Z}^r) \mapsto (\mathsf{X}^r + \mathsf{W}^r \odot \mathsf{Y}^r, \mathsf{Y}^r, \mathsf{Z}^r + \mathsf{X}^r \odot \mathsf{Y}^r + \mathsf{W}^r \odot \mathsf{Y}^r).$$

Again, for each $i \in \{0, \ldots, 31\}$, we have

$$\mathsf{Coe}_{\mathsf{Z}_i^r + \mathsf{X}_i^r \odot \mathsf{Y}_i^r + \mathsf{W}_i^r \odot \mathsf{Y}_i^r}(\boldsymbol{m}) = \mathsf{Coe}_{\mathsf{Z}_i^r}(\boldsymbol{m}) + \mathsf{Coe}_{\mathsf{X}_i^r \odot \mathsf{Y}_i^r}(\boldsymbol{m}) + \mathsf{Coe}_{\mathsf{W}_i^r \odot \mathsf{Y}_i^r}(\boldsymbol{m}) \tag{17}$$

By conditions (b) and (c), (17) reduces to

$$\mathsf{Coe}_{\mathsf{Z}_i^r + \mathsf{X}_i^r \odot \mathsf{Y}_i^r + \mathsf{W}_i^r \odot \mathsf{Y}_i^r}(\boldsymbol{m}) = \mathsf{Coe}_{\mathsf{W}_i^r \odot \mathsf{Y}_i^r}(\boldsymbol{m}). \tag{18}$$

Now, the same linear layer $L_{r \bmod 4}$ is applied to each word of the state, and (16) and (18) hold for all i. Thus, we have

$$L_{r \bmod 4} \times \begin{pmatrix} \mathsf{Coe}_{\mathsf{W}_0^r \odot \mathsf{Y}_0^r + \mathsf{X}_0^r}(\boldsymbol{m}) \\ \vdots \\ \mathsf{Coe}_{\mathsf{W}_{31}^r \odot \mathsf{Y}_{31}^r + \mathsf{X}_{31}^r}(\boldsymbol{m}) \end{pmatrix} = L_{r \bmod 4} \times \begin{pmatrix} \mathsf{Coe}_{\mathsf{Z}_0^r + \mathsf{X}_0^r \odot \mathsf{Y}_0^r + \mathsf{W}_0^r \odot \mathsf{Y}_0^r}(\boldsymbol{m}) \\ \vdots \\ \mathsf{Coe}_{\mathsf{Z}_{31}^r + \mathsf{X}_{31}^r \odot \mathsf{Y}_{31}^r + \mathsf{W}_{31}^r \odot \mathsf{Y}_{31}^r}(\boldsymbol{m}) \end{pmatrix}$$

From the above matrix product, it is evident that $\mathsf{Coe}_{\mathsf{X}_i^{r+1}}(\boldsymbol{m}) = \mathsf{Coe}_{\mathsf{Z}_i^{r+1}}(\boldsymbol{m})$ for all $i = 0, \ldots, 31$. This completes the proof. □

Remark 2. Theorem 1 implies that the monomial $\boldsymbol{m}$ is missing in the ANFs of $\mathsf{X}_i^r, \mathsf{Z}_i^r, \mathsf{X}_i^r \odot \mathsf{Y}_i^r$ and the coefficients of monomial $\boldsymbol{m}$ in X_i^{r+1} and Z_i^{r+1} are equal, for each $i \in \{0, 1, \ldots, 31\}$.

We emphasize that the observation in Theorem 1 could be destroyed easily given the ARADI's designers have used different linear mappings for each row. However, even different linear mappings do not fully prevent from having similar structural distinguishers. We summarize this observation in Corollary 1.

Corollary 1. *Let L_w, L_x, L_y, L_z be four different 32-bit to 32-bit invertible linear mappings and applied on words $\mathsf{W}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}$, respectively. Suppose the conditions in Theorem 1 are satisfied. Then, the following holds.*

$$L_x^{-1} \times \begin{pmatrix} \mathsf{Coe}_{\mathsf{X}_0^{r+1}}(\boldsymbol{m}) \\ \vdots \\ \mathsf{Coe}_{\mathsf{X}_{31}^{r+1}}(\boldsymbol{m}) \end{pmatrix} = L_y^{-1} \times \begin{pmatrix} \mathsf{Coe}_{\mathsf{Z}_0^{r+1}}(\boldsymbol{m}) \\ \vdots \\ \mathsf{Coe}_{\mathsf{Z}_{31}^{r}}(\boldsymbol{m}) \end{pmatrix}$$

Proof. The proof is similar to the proof of Theorem 1. We simply replace the fixed linear layer $L_{r \bmod 4}$ by L_x and L_z.

Application of Theorem 1. By applying Theorem 1, we can extend an integral distinguisher of r rounds to an integral distinguisher for $r+1$ rounds, provided we identify indices sets $\mathcal{I}_{\mathsf{w}}$, $\mathcal{I}_{\mathsf{x}}$, $\mathcal{I}_{\mathsf{y}}$, $\mathcal{I}_{\mathsf{z}}$ for which the conditions mentioned in the theorem are satisfied. The data complexity of the extended distinguisher remains the same, as it utilizes the same cube.

5 Structural Distinguishers of ARADI

In this section, we present distinguishers of round reduced ARADI using Theorem 1. First, we explain the basic idea of constructing indices sets $\mathcal{I}_{\mathsf{w}}$, $\mathcal{I}_{\mathsf{x}}$, $\mathcal{I}_{\mathsf{y}}$, $\mathcal{I}_{\mathsf{z}}$ for cube variables and then provide experimental results for up to 8 rounds ARADI. In the end, we give another interesting experimental distinguisher for 5 rounds.

5.1 Construction of Cube Indices Set

We use Remark 2 to find indices sets $\mathcal{I}_{\mathsf{w}}$, $\mathcal{I}_{\mathsf{x}}$, $\mathcal{I}_{\mathsf{y}}$, $\mathcal{I}_{\mathsf{z}}$ satisfying Theorem 1. We construct these sets in three steps as follows. Note that we use superscript to denote these sets at each step and consider all $i \in \{0, 1, \ldots, 31\}$.

1. Choose $\mathcal{I}^1_{\mathsf{w}}, \mathcal{I}^1_{\mathsf{x}}, \mathcal{I}^1_{\mathsf{y}}, \mathcal{I}^1_{\mathsf{z}}$ such that $d_1 = |\mathcal{I}^1_{\mathsf{w}}|+|\mathcal{I}^1_{\mathsf{x}}|+|\mathcal{I}^1_{\mathsf{y}}|+|\mathcal{I}^1_{\mathsf{z}}|$ and the algebraic degree of monomial
$$\boldsymbol{m}_1 = \prod_{i \,\in\, \mathcal{I}^1_{\mathsf{w}}} \mathsf{W}^0_i \cdot \prod_{i \,\in\, \mathcal{I}^1_{\mathsf{x}}} \mathsf{X}^0_i \cdot \prod_{i \,\in\, \mathcal{I}^1_{\mathsf{y}}} \mathsf{Y}^0_i \cdot \prod_{i \,\in\, \mathcal{I}^1_{\mathsf{z}}} \mathsf{Z}^0_i \,.$$
in X^r_i is at most $d_1 - 1$. If the algebraic degrees of $\boldsymbol{m}_1$ in both Z^r_i and $\mathsf{X}^r_i \odot \mathsf{Y}^r_i$ are also at most $d_1 - 1$, then we are done and we set $\mathcal{I}_{\mathsf{w}} = \mathcal{I}^1_{\mathsf{w}}$, $\mathcal{I}_{\mathsf{x}} = \mathcal{I}^1_{\mathsf{x}}$, $\mathcal{I}_{\mathsf{y}} = \mathcal{I}^1_{\mathsf{y}}$ and $\mathcal{I}_{\mathsf{z}} = \mathcal{I}^1_{\mathsf{z}}$. Else, we move on to the next step.
2. We update $(\mathcal{I}^1_{\mathsf{w}}, \mathcal{I}^1_{\mathsf{x}}, \mathcal{I}^1_{\mathsf{y}}, \mathcal{I}^1_{\mathsf{z}})$ to $(\mathcal{I}^2_{\mathsf{w}}, \mathcal{I}^2_{\mathsf{x}}, \mathcal{I}^2_{\mathsf{y}}, \mathcal{I}^2_{\mathsf{z}})$ by adding more indices so that (i) $d_2 = |\mathcal{I}^2_{\mathsf{w}}|+|\mathcal{I}^2_{\mathsf{x}}|+|\mathcal{I}^2_{\mathsf{y}}|+|\mathcal{I}^2_{\mathsf{z}}|$, (ii) $d_1 < d_2 < 127$, and (iii) the algebraic degree of monomial
$$\boldsymbol{m}_2 = \prod_{i \,\in\, \mathcal{I}^2_{\mathsf{w}}} \mathsf{W}^0_i \cdot \prod_{i \,\in\, \mathcal{I}^2_{\mathsf{x}}} \mathsf{X}^0_i \cdot \prod_{i \,\in\, \mathcal{I}^2_{\mathsf{y}}} \mathsf{Y}^0_i \cdot \prod_{i \,\in\, \mathcal{I}^2_{\mathsf{z}}} \mathsf{Z}^0_i \,.$$
in X^r_i and Z^r_i is at most $d_2 - 1$. If the algebraic degree of $\boldsymbol{m}_2$ in $\mathsf{X}^r_i \odot \mathsf{Y}^r_i$ is also at most $d_2 - 1$, then we are done and we set $\mathcal{I}_{\mathsf{w}} = \mathcal{I}^2_{\mathsf{w}}$, $\mathcal{I}_{\mathsf{x}} = \mathcal{I}^2_{\mathsf{x}}$, $\mathcal{I}_{\mathsf{y}} = \mathcal{I}^2_{\mathsf{y}}$ and $\mathcal{I}_{\mathsf{z}} = \mathcal{I}^2_{\mathsf{z}}$. Else, we proceed to the next step.
3. We update $(\mathcal{I}^2_{\mathsf{w}}, \mathcal{I}^2_{\mathsf{x}}, \mathcal{I}^2_{\mathsf{y}}, \mathcal{I}^2_{\mathsf{z}})$ to $(\mathcal{I}^3_{\mathsf{w}}, \mathcal{I}^3_{\mathsf{x}}, \mathcal{I}^3_{\mathsf{y}}, \mathcal{I}^3_{\mathsf{z}})$ by adding more indices so that (i) $d_3 = |\mathcal{I}^3_{\mathsf{w}}|+|\mathcal{I}^3_{\mathsf{x}}|+|\mathcal{I}^3_{\mathsf{y}}|+|\mathcal{I}^3_{\mathsf{z}}|$, (ii) $d_2 < d_3 < 127$, and (iii) the algebraic degree of monomial
$$\boldsymbol{m}_3 = \prod_{i \,\in\, \mathcal{I}^3_{\mathsf{w}}} \mathsf{W}^0_i \cdot \prod_{i \,\in\, \mathcal{I}^3_{\mathsf{x}}} \mathsf{X}^0_i \cdot \prod_{i \,\in\, \mathcal{I}^3_{\mathsf{y}}} \mathsf{Y}^0_i \cdot \prod_{i \,\in\, \mathcal{I}^3_{\mathsf{z}}} \mathsf{Z}^0_i \,.$$

in $\mathsf{X}_i^r \odot \mathsf{Y}_i^r$ is at most $d_3 - 1$. Finally, we set $\mathcal{I}_\mathsf{w} = \mathcal{I}_\mathsf{w}^3$, $\mathcal{I}_\mathsf{x} = \mathcal{I}_\mathsf{x}^3$, $\mathcal{I}_\mathsf{y} = \mathcal{I}_\mathsf{y}^3$ and $\mathcal{I}_\mathsf{z} = \mathcal{I}_\mathsf{z}^3$.

Note that we use the division property to compute the algebraic degrees in the above three steps (Section 3).

5.2 Experimental Results

We use the aforementioned approach and give the concrete cube indices sets with their respective algebraic degrees in Table 4. By Theorem 1, each of the r-round distinguisher in Table 4 can be easily extended to one additional round.

Table 4. Indices set and the corresponding algebraic degrees for up to 7 rounds. We evaluate the degrees for each $i \in \{0, 1, \ldots, 31\}$ and the table reports the maximum degree out of all 32 bits.

Round r	Indices	Degree of monomial $\boldsymbol{m}$ in X_i^r	Z_i^r	$\mathsf{X}_i^r \odot \mathsf{Y}_i^r$	Cube dimension
3	$\mathcal{I}_\mathsf{W} = \{28, \ldots, 31\}$ $\mathcal{I}_\mathsf{X} = \phi$ $\mathcal{I}_\mathsf{Y} = \phi$ $\mathcal{I}_\mathsf{Z} = \phi$	2	2	3	4
4	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 15\}$ $\mathcal{I}_\mathsf{X} = \phi$ $\mathcal{I}_\mathsf{Y} = \phi$ $\mathcal{I}_\mathsf{Z} = \phi$	9	9	15	16
5	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 20\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 20\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 20\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 20\}$	70	70	83	84
6	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 28\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 27\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 27\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 27\}$	105	105	112	113
7	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 30\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 30\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 30\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 30\}$	120	120	123	124

Again the values in Table

We emphasize that the values in Table 4 represent the upper bounds, and they may improve on further investigations. Moreover, instead of considering all 32 bits, if we focus on a few specific bits, then the cube dimensions (or the data complexity) can be further reduced.

5.3 Another 5-round Distinguisher

In Table 5, we report a 5-round distinguisher with data complexity 2^{13}. A notable property of this distinguisher is that consecutive bytes in the cube-sum values of X^5 and Z^5 are equal. On the other hand, the 5-round distinguisher in Table 3 requires 2^{16} data and has cube-sum values of complete X^5 and Z^5 equal.

Table 5. A 5-round distinguisher of ARADI with equal cube-sum values (marked in blue and red) after 5 rounds. Here $\mathcal{I}_{\mathsf{w}} = \{11, 12, \ldots, 23\}$, $\mathcal{I}_{\mathsf{x}} = \phi$, $\mathcal{I}_{\mathsf{y}} = \phi$ and $\mathcal{I}_{\mathsf{z}} = \phi$. The plaintext bits corresponding to $\tilde{\mathcal{I}}_{\mathsf{w}}$, $\tilde{\mathcal{I}}_{\mathsf{x}}$, $\tilde{\mathcal{I}}_{\mathsf{y}}$, $\tilde{\mathcal{I}}_{\mathsf{z}}$ are set as constant zero.

Master key	Cube-sum values in word W^5	X^5	Y^5	Z^5
f9bd8218 8f19ef37 12e12e0b b0ebb077 585c1931 abd53e97 b358afdc dca17a08	924df92e	84848282	43ec091e	81813636
3fd88155 55c6f9e2 aec73cd7 57ddd4f0 cbcaed87 c34cbe54 fe3a8373 b47e6f3a	acdcd029	cfcfc6c6	84ca03d4	8d8d8787
d2da431e 711c7414 8eef3216 290f6caa 4ad19d3b 9202f74c e2c558ea e1753797	d3293e50	16169a9a	ca7f832e	0a0a6969
4c98b38d 70bb8b6a 3fc579bd a23ea6b2 d8abc0e2 7d624fd5 2a7f6de6 9bc44e5e	2e628d14	bdbd4141	19a52f08	9d9dc5c5

Remark 3. We believe the distinguishing property of Table 5 can also be observed for higher rounds. However, due to larger dimensions of cubes and limited computational resources, we could not find any practical distinguishers for 6 or more rounds.

6 Impossibility of Fixing Structural Distinguishers

In this section, we investigate whether avoiding the structural patterns of ARADI as given in Sections 4 and 5 is possible. We discuss whether changing the linear layer or the order of composition of Toffoli gates could avoid these patterns. Ultimately, we answer the posed question negatively based on our experimental results.

6.1 Changing the Linear Layer

The linear layer of ARADI applies the same mapping $L_{r \bmod 4}$ to each word of the state at r-th round. This is one of the major reasons why such structural patterns are clearly noticed. So, having different linear mappings for each row may break these patterns. However, we can still observe new deterministic patterns by applying Corollary 1. So, the crucial observation here is that the structural patterns of ARADI (in Sections 4 and 5) are independent of the linear layer.

6.2 Changing the Composition of Toffoli Gates

The nonlinear layer of ARADI applies 4 Toffoli gates to words $\mathsf{W}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}$ in a specific sequence (see Section 2.3). In total there are $\binom{4}{2}$ possible ways the first Toffoli gate can be computed, i.e., by taking bitwise AND of either of $\mathsf{W} \odot \mathsf{X}$, $\mathsf{W} \odot \mathsf{Y}$, $\mathsf{W} \odot \mathsf{Z}$, $\mathsf{X} \odot \mathsf{Y}$, $\mathsf{X} \odot \mathsf{Z}$ and $\mathsf{Y} \odot \mathsf{Z}$.

We compute the algebraic degree growth for each product term for up to 7 rounds in Table 6. We notice that the degrees are consistently less than the cube dimension after each round (except for $\mathsf{W}_i^r \odot \mathsf{Y}_i^r$, for which the degree drops in round 7). Hence, from Table 6, it is evident that if we change the order in which the state words are updated without altering the word combination for the AND operation, the cube-sum of the first updated word will be zero. Thus, similar to Theorem 1, in this case as well, we can extend the distinguishers from r rounds to $r + 1$ rounds.

In summary, irrespective of which product term we select out of the 6 possible combinations for the first Toffoli gate, we still have observations similar to Theorem 1. Thus, the main takeaway is that ARADI's nonlinear layer needs to be re-designed to prevent these structural distinguishes, which are not noticeable in other state-of-the-art block ciphers.

7 Key Recovery Attacks

This section presents a key recovery attack on 10 rounds ARADI. We first provide some observations related to the key-scheduling of ARADI, and then give the details of key recovery attack.

7.1 Observation on ARADI Key Schedule

Let $rk^0, rk^1, \ldots, rk^r$ denote the 128-bit round keys for r rounds ARADI. Further, rk_j^i denotes the j-th 32-bit word of rk^i, for $j \in \{0, 1, 2, 3\}$ and $i = 0, \ldots, r$. Moreover, by $rk_j^i[k]$ we mean the k-th bit of rk_j^i. Then, for any i, the following holds for the ARADI key schedule algorithm.

1. $rk_1^{i+2} = \mathsf{truncate}(M_0(rk_0^{i+1}, rk_1^{i+1}), 32)$
2. $rk_3^{i+2} = \mathsf{truncate}(M_1(rk_2^{i+1}, rk_3^{i+1}), 32)$

Here M_0 and M_1 are two 64-bit to 64-bit linear maps (see (9)) and $\mathsf{truncate}(\cdot, 32)$ gives the first 32 bits of the output of M_0 or M_1.

The above observation shows that two keywords of round $i + 2$ can be fully determined from the keywords of round $i + 1$. This means for rounds $i + 1$ and $i + 2$, the entropy of round keys rk^{i+1} and rk^{i+2} is 192 bits rather than 256 bits. However, recovering rk^{i+1} and rk^{i+2} is not equivalent to recovering the master key. This is because two additional intermediate keywords must be guessed to invert the key schedule.

Table 6. Index sets and the corresponding algebraic degrees for up to 7 rounds for all possible AND combinations for the first Toffoli gate. We evaluate the degrees for each $i \in \{0, 1, \ldots, 31\}$, and the table reports the maximum degree out of all 32 bits.

Round r	Indices	Degree of monomial $\boldsymbol{m}$ in						Cube dimension
		$\mathsf{X}_i^r \odot \mathsf{Y}_i^r$	$\mathsf{W}_i^r \odot \mathsf{X}_i^r$	$\mathsf{W}_i^r \odot \mathsf{Y}_i^r$	$\mathsf{W}_i^r \odot \mathsf{Z}_i^r$	$\mathsf{X}_i^r \odot \mathsf{Z}_i^r$	$\mathsf{Y}_i^r \odot \mathsf{Z}_i^r$	
3	$\mathcal{I}_\mathsf{W} = \{28, \ldots, 31\}$ $\mathcal{I}_\mathsf{X} = \phi$ $\mathcal{I}_\mathsf{Y} = \phi$ $\mathcal{I}_\mathsf{Z} = \phi$	3	3	3	3	3	3	4
4	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 15\}$ $\mathcal{I}_\mathsf{X} = \phi$ $\mathcal{I}_\mathsf{Y} = \phi$ $\mathcal{I}_\mathsf{Z} = \phi$	15	15	16	15	14	15	16
5	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 20\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 20\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 20\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 20\}$	83	83	84	83	80	83	84
6	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 28\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 27\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 27\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 27\}$	112	112	113	112	112	112	113
7	$\mathcal{I}_\mathsf{W} = \{0, \ldots, 30\}$ $\mathcal{I}_\mathsf{X} = \{0, \ldots, 30\}$ $\mathcal{I}_\mathsf{Y} = \{0, \ldots, 30\}$ $\mathcal{I}_\mathsf{Z} = \{0, \ldots, 30\}$	123	123	123	123	123	123	124

7.2 A Key Recovery Attack on 10-round ARADI

Figure 1 shows the high-level overview of the 10-round key recovery attack on ARADI. In the figure, the words $\mathsf{W}, \mathsf{X}, \mathsf{Y}$ and Z are shown by rows 0, 1, 2 and 3, respectively.

We use the 8-round integral distinguisher (cube dimension 124, Table 2) in our attack. So, the cube-sum after 8 rounds is zero in each cell (shown in green in Figure 1). We add 2 rounds to this distinguisher to mount 10-round key recovery attack. In our attack, we use the fact that each column of the state after 2 rounds of partial decryption depends on 3 columns of 9-th round state and 9 columns of 10-th round state. For instance, the bits in column 0 after 8-th round key addition depend on columns $\{0, 5, 24\}$ of 9-th round and columns $\{0, 5, 6, 11, 13, 23, 24, 28, 30\}$ of 10-th round (shown with gray cells). Accordingly, if we guess round key bits corresponding to these columns we can compute the zero-th column after the round 8 key addition. This procedure is shown in the figure for column 0.

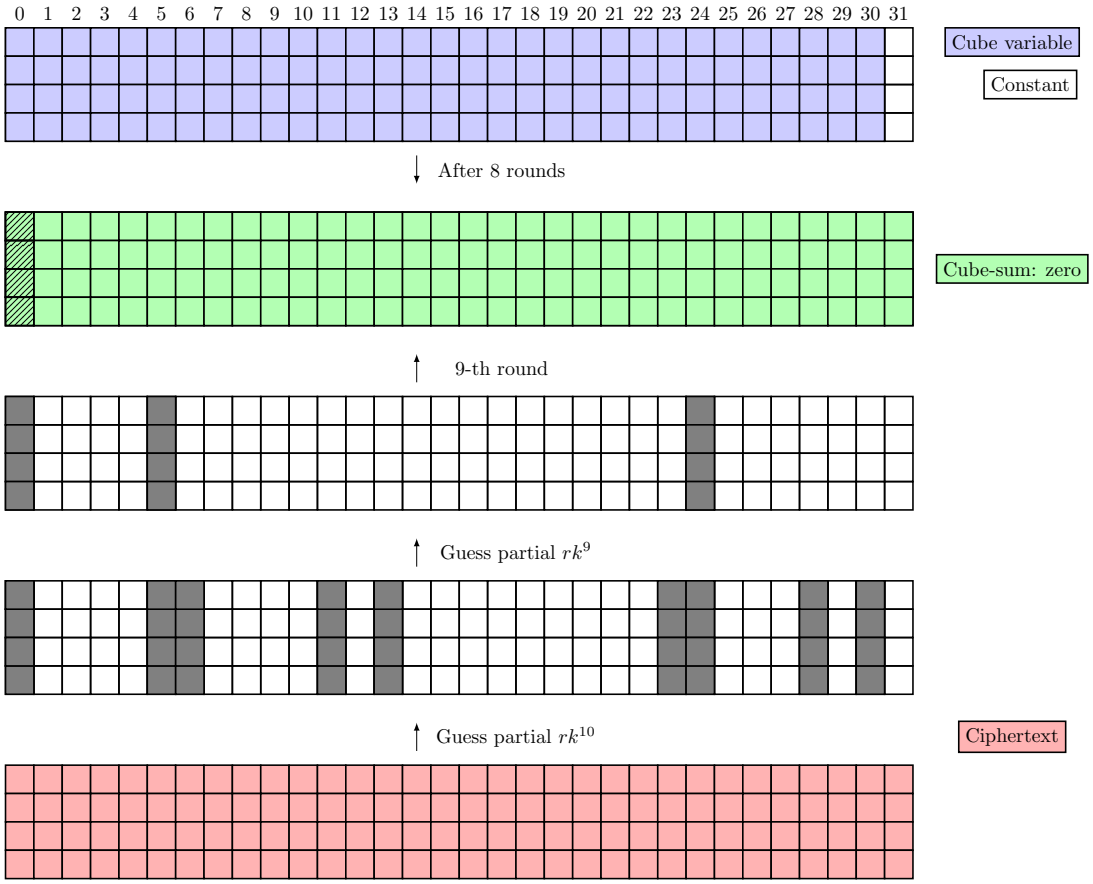

Fig. 1. An overview of 10-round key recovery attack on ARADI. After partial decryption for 2 rounds, matching is done at positions as shown by green squares with lines (column 0). The gray boxes highlight the round key bits which needs to be guessed to compute the respective state columns.

We now explain the detailed attack steps along with their respective complexities for column 0 only. We first define $\mathcal{SK}$ and $\mathcal{CT}$ as the set of round key bits and ciphertext bits which are needed to partially decrypt two rounds. More

precisely, we have

$$\begin{aligned}\mathcal{SK} = \{&rk_0^9[0], rk_1^9[0], rk_2^9[0], rk_3^9[0], rk_0^9[5], rk_1^9[5], rk_2^9[5], rk_3^9[5],\\ &rk_0^9[24], rk_1^9[24], rk_2^9[24], rk_3^9[24], rk_0^{10}[0], rk_1^{10}[0], rk_2^{10}[0], rk_3^{10}[0],\\ &rk_0^{10}[0], rk_1^{10}[0], rk_2^{10}[0], rk_3^{10}[0], rk_0^{10}[5], rk_1^{10}[5], rk_2^{10}[5], rk_3^{10}[5],\\ &rk_0^{10}[6], rk_1^{10}[6], rk_2^{10}[6], rk_3^{10}[6], rk_0^{10}[11], rk_1^{10}[11], rk_2^{10}[11], rk_3^{10}[11],\\ &rk_0^{10}[13], rk_1^{10}[13], rk_2^{10}[13], rk_3^{10}[13], rk_0^{10}[23], rk_1^{10}[23], rk_2^{10}[23], rk_3^{10}[23],\\ &rk_0^{10}[24], rk_1^{10}[24], rk_2^{10}[24], rk_3^{10}[24], rk_0^{10}[28], rk_1^{10}[28], rk_2^{10}[28], rk_3^{10}[28],\\ &rk_0^{10}[30], rk_1^{10}[30], rk_2^{10}[30], rk_3^{10}[30]\ \},\end{aligned}$$

and

$$\begin{aligned}\mathcal{CT} = \{&\mathsf{W}_0^{10}, \mathsf{W}_5^{10}, \mathsf{W}_6^{10}, \mathsf{W}_{11}^{10}, \mathsf{W}_{13}^{10}, \mathsf{W}_{23}^{10}, \mathsf{W}_{24}^{10}, \mathsf{W}_{28}^{10}, \mathsf{W}_{30}^{10},\\ &\mathsf{X}_0^{10}, \mathsf{X}_5^{10}, \mathsf{X}_6^{10}, \mathsf{X}_{11}^{10}, \mathsf{X}_{13}^{10}, \mathsf{X}_{23}^{10}, \mathsf{X}_{24}^{10}, \mathsf{X}_{28}^{10}, \mathsf{X}_{30}^{10},\\ &\mathsf{Y}_0^{10}, \mathsf{Y}_5^{10}, \mathsf{Y}_6^{10}, \mathsf{Y}_{11}^{10}, \mathsf{Y}_{13}^{10}, \mathsf{Y}_{23}^{10}, \mathsf{Y}_{24}^{10}, \mathsf{Y}_{28}^{10}, \mathsf{Y}_{30}^{10},\\ &\mathsf{Z}_0^{10}, \mathsf{Z}_5^{10}, \mathsf{Z}_6^{10}, \mathsf{Z}_{11}^{10}, \mathsf{Z}_{13}^{10}, \mathsf{Z}_{23}^{10}, \mathsf{Z}_{24}^{10}, \mathsf{Z}_{28}^{10}, \mathsf{Z}_{30}^{10}\ \}\,.\end{aligned}$$

Note that $|\mathcal{SK}| = 48$ and $|\mathcal{CT}| = 36$. The attack steps are as follows.

Step 1: Preparing plaintexts. We prepare a set of 2^{124} plaintexts with cube indices corresponding to the fifth row of Table 2. We denote this set by $\mathcal{P}$.

Step 2: Querying ARADI oracle and storing ciphertexts. For each $p \in \mathcal{P}$, we query 10-round ARADI oracle and store the ciphertexts in the set $\mathcal{C}$. This step requires 2^{124} encryption queries (1 query = 10-round ARADI), and $2^{124} \cdot 128$ bits of memory.

Step 3: Key recovery phase. For the zero-th column, we recover key bits as follows.

3.1 For each guess sk of $\mathcal{SK}$, we compute the values $\bigoplus \mathsf{W}_0^8, \bigoplus \mathsf{X}_0^8, \bigoplus \mathsf{Y}_0^8$ and $\bigoplus \mathsf{Z}_0^8$, by partially decrypting all 2^{124} ciphertexts. Note that for the partial decryption, we only need 36 bits of each ciphertext which are given by set $\mathcal{CT}$. This step requires $2^{48} \cdot 2^{124}$ 2-round decryptions.

3.2 If $\bigoplus \mathsf{W}_0^8 = 0, \bigoplus \mathsf{X}_0^8 = 0, \bigoplus \mathsf{Y}_0^8 = 0$ and $\bigoplus \mathsf{Z}_0^8 = 0$, we add sk as a possible 48-bit key candidate.

At the end of step 3, we reduce the key space from 2^{48} to 2^{44} since we are checking values at 4 positions.

Discussion on further filtering of key space. We repeat step 3 for other columns (by updating the sets $\mathcal{SK}$ and $\mathcal{CT}$ based on the column index), and considering the reduced key space and independent round key bits at each iteration. In the worst case, we have to repeat step 3 in total 32 times. Thus, the overall complexity to recover rk^9 and rk^{10} is then dominated by $32 \cdot 2^{48} \cdot 2^{124}$ 2-round decryptions.

Having recovered rk^9 and rk^{10}, the remaining 64 bits needed for inverting the key schedule (see Section 7.1) can be obtained by exhaustive search.

Attack complexities. Combining all the previous steps, the entire 10-round attack has the following complexities.

$$\begin{aligned} \text{Data} &= 2^{124} \\ \text{Memory} &= 2^{124} \cdot 128 = 2^{131} \text{ bits} \\ \text{Time} &= 2^{124} + 32 \cdot 2^{48} \cdot 2^{124} + 2^{64} \approx 2^{177} \end{aligned} \tag{19}$$

7.3 Improving Number of Rounds for Key Recovery

While the main goal of this work was to investigate the deterministic structural patterns of ARADI, which the direct application of division property could not obtain, we provided the 10-round key recovery attack for the sake of completeness. It is possible that by exploiting more properties of ARADI's key schedule, one may append three rounds to our 8 rounds distinguisher resulting in 11-round key recovery attack. However, based on our current analysis, we do not see a trivial way to achieve this. One of the main reason is we need to guess round key bits for at least 32 columns and the data complexity is too high for the distinguisher. We leave an analysis of adding more rounds for key recovery as a future work.

8 Conclusions

In this paper, we presented structural algebraic distinguishers for round-reduced ARADI (up to 8 rounds) by exploiting the composition of Toffoli gates in the round function. These distinguishers are difficult to obtain with the direct application of division property, and have lower data complexities than the existing algebraic distinguishers. We showed that the weakness is inherent to Toffoli gates, independent of the linear layer, and therefore, non-trivial to avoid. We also give a key recovery attack on 10 rounds, leaving the security margin of ARADI to only 6 rounds.

We believe the complexities of our distinguishers can be reduced as the provided degrees are merely the upper bounds. Similarly, the number of rounds can also be improved. Moreover, extension of Table 5 distinguishers (where two consecutive bytes are equal) to more rounds, and understanding its theoretical reasoning would be interesting. On an another note, exploiting ARADI's key-schedule to add more rounds for key recovery may be possible. We leave all these problems as a future work.

References

1. Anand, R., Banik, S., Caforio, A., Ishikawa, T., Isobe, T., Liu, F., Minematsu, K., Rahman, M., Sakamoto, K.: Gleeok: A family of low-latency prfs and its applications to authenticated encryption. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(2), 545–587 (2024). https://doi.org/10.46586/TCHES.V2024.I2.545-587, https://doi.org/10.46586/tches.v2024.i2.545-587
2. Aumasson, J., Dinur, I., Meier, W., Shamir, A.: Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In: Dunkelman, O. (ed.) Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5665, pp. 1–22. Springer (2009). https://doi.org/10.1007/978-3-642-03317-9_1, https://doi.org/10.1007/978-3-642-03317-9_1
3. Avanzi, R.: The QARMA Block Cipher Family. Almost MDS Matrices Over Rings With Zero Divisors, Nearly Symmetric Even-Mansour Constructions With Non-Involutory Central Rounds, and Search Heuristics for Low-Latency S-Boxes. IACR Trans. Symmetric Cryptol. **2017**(1), 4–44 (2017). https://doi.org/10.13154/TOSC.V2017.I1.4-44
4. Avanzi, R., Banik, S., Dunkelman, O., Eichlseder, M., Ghosh, S., Nageler, M., Regazzoni, F.: The QARMAv2 Family of Tweakable Block Ciphers. IACR Trans. Symmetric Cryptol. **2023**(3), 25–73 (2023). https://doi.org/10.46586/TOSC.V2023.I3.25-73
5. Avanzi, R., Dunkelman, O., Ghosh, S.: A note on ARADI and LLAMA. Cryptology ePrint Archive, Paper 2024/1328 (2024), https://eprint.iacr.org/2024/1328
6. Banik, S., Isobe, T., Liu, F., Minematsu, K., Sakamoto, K.: Orthros: A Low-Latency PRF. IACR Trans. Symmetric Cryptol. **2021**(1), 37–77 (2021). https://doi.org/10.46586/TOSC.V2021.I1.37-77
7. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016). https://doi.org/10.1007/978-3-662-53008-5_5
8. Belkheyar, Y., Daemen, J., Dobraunig, C., Ghosh, S., Rasoolzadeh, S.: BipBip: A Low-Latency Tweakable Block Cipher with Small Dimensions. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(1), 326–368 (2023). https://doi.org/10.46586/TCHES.V2023.I1.326-368
9. Belkheyar, Y., Daemen, J., Dobraunig, C., Ghosh, S., Rasoolzadeh, S.: Introducing two Low-Latency Cipher Families: Sonic and SuperSonic. IACR Cryptol. ePrint Arch. p. 878 (2023), https://eprint.iacr.org/2023/878
10. Bellini, E., Formenti, M., Gérault, D., Grados, J., Hambitzer, A., Huang, Y.J., Huynh, P., Rachidi, M., Rohit, R., Tiwari, S.K.: CLAASPing ARADI: Automated analysis of the ARADI block cipher. Cryptology ePrint Archive, Paper 2024/1324 (2024), https://eprint.iacr.org/2024/1324
11. Bellini, E., Gérault, D., Grados, J., Huang, Y.J., Makarim, R.H., Rachidi, M., Tiwari, S.K.: CLAASP: A Cryptographic Library for the Automated Analysis of Symmetric Primitives. In: Carlet, C., Mandal, K., Rijmen, V. (eds.) Selected Areas in Cryptography - SAC 2023 - 30th International Conference, Fredericton,

Canada, August 14-18, 2023, Revised Selected Papers. Lecture Notes in Computer Science, vol. 14201, pp. 387–408. Springer (2023). https://doi.org/10.1007/978-3-031-53368-6_19
12. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7658, pp. 208–225. Springer (2012). https://doi.org/10.1007/978-3-642-34961-4_14
13. Bozilov, D., Eichlseder, M., Knezevic, M., Lambin, B., Leander, G., Moos, T., Nikov, V., Rasoolzadeh, S., Todo, Y., Wiemer, F.: PRINCEv2 - More Security for (Almost) No Overhead. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12804, pp. 483–511. Springer (2020). https://doi.org/10.1007/978-3-030-81652-0_19
14. Canale, F., Güneysu, T., Leander, G., Thoma, J.P., Todo, Y., Ueno, R.: SCARF - A Low-Latency Block Cipher for Secure Cache-Randomization. In: Calandrino, J.A., Troncoso, C. (eds.) 32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023. pp. 1937–1954. USENIX Association (2023), https://www.usenix.org/conference/usenixsecurity23/presentation/canale
15. Carlet, C., Crama, Y., Hammer, P.L.: Boolean functions for cryptography and error-correcting codes. In: Crama, Y., Hammer, P.L. (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering, pp. 257–397. Cambridge University Press (2010). https://doi.org/10.1017/CBO9780511780448.011, https://doi.org/10.1017/cbo9780511780448.011
16. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher square. In: Biham, E. (ed.) Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1267, pp. 149–165. Springer (1997). https://doi.org/10.1007/BFB0052343, https://doi.org/10.1007/BFb0052343
17. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. pp. 278–299 (2009)
18. Greene, P., Motley, M., Weeks, B.: ARADI and LLAMA: Low-Latency Cryptography for Memory Encryption. IACR Cryptol. ePrint Arch. p. 1240 (2024), https://eprint.iacr.org/2024/1240
19. Hao, Y., Isobe, T., Jiao, L., Li, C., Meier, W., Todo, Y., Wang, Q.: Improved division property based cube attacks exploiting algebraic properties of superpoly. IEEE Trans. Computers **68**(10), 1470–1486 (2019). https://doi.org/10.1109/TC.2019.2909871, https://doi.org/10.1109/TC.2019.2909871
20. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for Three-Subset Division Property Without Unknown Subset - Improved Cube Attacks Against Trivium and Grain-128AEAD. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020,

Proceedings, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 466–495. Springer (2020). https://doi.org/10.1007/978-3-030-45721-1_17
21. Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Lower bounds on the degree of block ciphers. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12491, pp. 537–566. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_18, https://doi.org/10.1007/978-3-030-64837-4_18
22. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12491, pp. 446–476. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_15, https://doi.org/10.1007/978-3-030-64837-4_15
23. Knudsen, L.R., Wagner, D.A.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2365, pp. 112–127. Springer (2002). https://doi.org/10.1007/3-540-45661-9_9
24. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography: Two Sides of One Tapestry. pp. 227–233. Springer US, Boston, MA (1994). https://doi.org/10.1007/978-1-4615-2694-0_23
25. Leander, G., Moos, T., Moradi, A., Rasoolzadeh, S.: The SPEEDY Family of Block Ciphers Engineering an Ultra Low-Latency Cipher from Gate Level for Secure Processor Architectures. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(4), 510–545 (2021). https://doi.org/10.46586/TCHES.V2021.I4.510-545
26. Todo, Y.: Structural Evaluation by Generalized Integral Property. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 287–314. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_12
27. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 250–279. Springer (2017). https://doi.org/10.1007/978-3-319-63697-9_9, https://doi.org/10.1007/978-3-319-63697-9_9
28. Todo, Y., Morii, M.: Bit-Based Division Property and Application to Simon Family. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 357–377. Springer (2016). https://doi.org/10.1007/978-3-662-52993-5_18
29. Wang, J., Huang, T., Wu, S., Liu, Z.: Twinkle: A family of low-latency schemes for authenticated encryption and pointer authentication. IACR Communications in Cryptology **1**(2) (2024). https://doi.org/10.62056/a3n59qgxq

30. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: Milp-aided method of searching division property using three subsets and applications. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 398–427. Springer (2019). https://doi.org/10.1007/978-3-030-34618-8_14, https://doi.org/10.1007/978-3-030-34618-8_14
31. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 648–678 (2016). https://doi.org/10.1007/978-3-662-53887-6_24, https://doi.org/10.1007/978-3-662-53887-6_24

A MILP Models for Underlying Components of ARADI

Algorithm 3: MILP model for RoundkeyXOR

Input: $\mathcal{M}, [a_0, \ldots, a_{31}], [b_0, \ldots, b_{31}], [c_0, \ldots, c_{31}]$

for $i = 0$ *to* 31 **do**

  $\mathcal{M}.addConstr \leftarrow a_i = b_i + c_i$

return $\mathcal{M}$

Algorithm 4: MILP model for S-box

Input: $\mathcal{M}, [b_0, \ldots, b_{31}], [c_0, \ldots, c_{31}], [d_0, \ldots, d_{31}], [e_0, \ldots, e_{31}], [a_0, \ldots, a_{127}]$

for $i = 0$ *to* 31 **do**

  $\mathsf{SB}(\mathcal{M}, [b_i, c_i, d_i, e_i], [a_i, a_{i+32}, a_{i+64}, a_{i+96}])$

return $\mathcal{M}$

Algorithm 5: MILP model for SB

Input: $\mathcal{M}, [x_0, \ldots, x_3], [y_0, \ldots, y_3]$

$\mathcal{M}.addVar \leftarrow p_i$, for $i = 0, \ldots, 7$
$\mathcal{M}.addVar \leftarrow q_i$, for $i = 0, \ldots, 4$
$\mathcal{M}.addVar \leftarrow r_i$, for $i = 0, \ldots, 8$
$\mathcal{M}.addVar \leftarrow s_i$, for $i = 0, \ldots, 3$
$\mathcal{M}.addVar \leftarrow t_i$, for $i = 0, \ldots, 9$

$\mathcal{M}.addConstr \leftarrow x_0 \geq p_i$, for $i = 0, \ldots, 7$; $\mathcal{M}.addConstr \leftarrow \sum_{i=0}^{7} p_i \geq x_0$
$\mathcal{M}.addConstr \leftarrow x_1 \geq q_i$, for $i = 0, \ldots, 4$; $\mathcal{M}.addConstr \leftarrow \sum_{i=0}^{4} q_i \geq x_1$
$\mathcal{M}.addConstr \leftarrow x_2 \geq r_i$, for $i = 0, \ldots, 8$; $\mathcal{M}.addConstr \leftarrow \sum_{i=0}^{8} r_i \geq x_2$
$\mathcal{M}.addConstr \leftarrow x_3 \geq s_i$, for $i = 0, \ldots, 3$; $\mathcal{M}.addConstr \leftarrow \sum_{i=0}^{3} s_i \geq x_3$

$\mathcal{M}.addConstr \leftarrow t_0 = p_0$; $\mathcal{M}.addConstr \leftarrow t_0 = r_0$; $\mathcal{M}.addConstr \leftarrow t_0 = s_0$
$\mathcal{M}.addConstr \leftarrow t_1 = p_1$; $\mathcal{M}.addConstr \leftarrow t_1 = r_1$;
$\mathcal{M}.addConstr \leftarrow t_2 = q_0$; $\mathcal{M}.addConstr \leftarrow t_2 = r_2$;
$\mathcal{M}.addConstr \leftarrow t_3 = q_1$; $\mathcal{M}.addConstr \leftarrow t_3 = s_1$;
$\mathcal{M}.addConstr \leftarrow y_0 = t_0 + t_1 + p_2 + t_2 + t_3$;

$\mathcal{M}.addConstr \leftarrow t_4 = p_3$; $\mathcal{M}.addConstr \leftarrow t_3 = r_3$;
$\mathcal{M}.addConstr \leftarrow y_1 = t_4 + q_2$;

$\mathcal{M}.addConstr \leftarrow t_5 = p_4$; $\mathcal{M}.addConstr \leftarrow t_5 = q_3$; $\mathcal{M}.addConstr \leftarrow t_5 = r_4$
$\mathcal{M}.addConstr \leftarrow t_6 = p_5$; $\mathcal{M}.addConstr \leftarrow t_6 = r_5$;
$\mathcal{M}.addConstr \leftarrow t_7 = p_6$; $\mathcal{M}.addConstr \leftarrow t_7 = s_2$;
$\mathcal{M}.addConstr \leftarrow y_2 = t_5 + t_6 + t_7 + r_6$;

$\mathcal{M}.addConstr \leftarrow t_8 = p_7$; $\mathcal{M}.addConstr \leftarrow t_8 = r_7$;
$\mathcal{M}.addConstr \leftarrow t_9 = q_4$; $\mathcal{M}.addConstr \leftarrow t_9 = r_8$;
$\mathcal{M}.addConstr \leftarrow y_3 = t_8 + t_9 + s_4$;

return $\mathcal{M}$

Algorithm 6: MILP model for Linear-map

Input: $\mathcal{M}, j, [x_0, \ldots, x_{31}], [y_0, \ldots, y_{31}]$

$a = [11, 10, 9, 8]$; $b = [8, 9, 4, 9]$; $c = [14, 11, 14, 7]$
for $i = 0$ *to* 31 **do**
  $\mathcal{M}.addVar \leftarrow u_i$, $\mathcal{M}.addVar \leftarrow v_i$, $\mathcal{M}.addVar \leftarrow w_i$
  $\mathcal{M}.addConstr \leftarrow x_i \geq u_i$
  $\mathcal{M}.addConstr \leftarrow x_i \geq v_i$
  $\mathcal{M}.addConstr \leftarrow x_i \geq w_i$
  $\mathcal{M}.addConstr \leftarrow u_i + v_i + w_i \geq x_i$
$u^l = u_0 \| \cdots \| u_{15}$; $u^r = u_{16} \| \cdots \| u_{31}$
$v^l = v_0 \| \cdots \| v_{15}$; $v^r = v_{16} \| \cdots \| v_{31}$
$w^l = w_0 \| \cdots \| w_{15}$; $w^r = w_{16} \| \cdots \| w_{31}$
$v^l \leftarrow$ Circular-shift$(v^l, a[j])$
$w^r \leftarrow$ Circular-shift$(w^r, c[j])$
for $i = 0$ *to* 15 **do**
  $\mathcal{M}.addVar \leftarrow L_i$
  $\mathcal{M}.addConstr \leftarrow L_i = u_i^l + v_i^l$
  $\mathcal{M}.addConstr \leftarrow y_i = L_i + w_i^r$
$v^r \leftarrow$ Circular-shift$(v^r, a[j])$
$w^l \leftarrow$ Circular-shift$(w^l, b[j])$
for $i = 0$ *to* 15 **do**
  $\mathcal{M}.addVar \leftarrow R_i$
  $\mathcal{M}.addConstr \leftarrow R_i = u_i^r + v_i^r$
  $\mathcal{M}.addConstr \leftarrow y_{i+16} = R_i + w_i^l$
return $\mathcal{M}$

Algorithm 7: Circular-shift

Input: 16-bit vector u, integer i (< 16)
Output: 16-bit vector

$u_{\text{shifted}} \leftarrow (u \ll i) \mid (u \gg (16 - i))$
return u_{shifted}