Byte-wise equal property of ARADI

Sunyeop Kim[1], Insung Kim[1], Dongjae Lee[4], Deukjo Hong[3], Jaechul Sung[2], and Seokhie Hong[1]

[1] Institute of Cyber Security & Privacy (ICSP), Korea University, South Korea
kin3548@gmail.com, cmcom35@korea.ac.kr, shhong@korea.ac.kr
[2] Department of Mathematics, University of Seoul, South Korea jcsung@uos.ac.kr
[3] Department of Information Technology & Engineering, Jeonbuk National University, South Korea deukjo.hong@jbnu.ac.kr
[4] Department of Convergence Security, Kangwon National University, South Korea dongjae.lee@kangwon.ac.kr

Abstract. ARADI is a low-latency block cipher proposed by the NSA (National Security Agency) in 2024 for memory encryption. Bellini et al. experimentally demonstrated that in specific cubes of 5-round ARADI, the cube sums are byte-wise equal, for example, to 0x9d9dc5c5. This paper modifies the MILP-based division property algorithm to prove this and observes that the rotation amount of 8 in ARADI causes cancellations of monomials, allowing us to extend the byte-wise equal property up to 8 rounds. As a result, we obtained distinguishers for rounds 6 and 7 with lower data complexities of 2^{77} and 2^{112}, respectively, compared to previous methods.

Keywords: Block cipher, Integral attack, Algebraic attack

1 Introduction

In modern processor architectures, CPUs are often protected within a secure enclave, ensuring security. However, it is practically impossible to provide the same level of protection for RAM. For this reason, it is necessary to encrypt data recorded in RAM using a secret key. In this case, encryption must be performed with very low latency to avoid delaying memory access.

To address this, in 2024, the U.S. National Security Agency (NSA) proposed a low-latency block cipher called ARADI, which is based on Toffoli gates and has a 128-bit block size and a 256-bit key size [4]. Unlike other symmetric key primitives, ARADI does not provide an explanation for the chosen structure, parameters, or security analysis. Therefore, to verify the security of ARADI, it is necessary to conduct a thorough review of various cryptographic analysis techniques.

Following the publication of ARADI, Bellini et al. [1] analyzed the ARADI cipher using an automated tool called CLAASP [2]. They applied various cryptographic analysis techniques to ARADI, resulting in statistical black-box analysis (avalanche and diffusion tests), differential and linear trails up to 9 rounds,

impossible differential trails up to 8 rounds, an integral distinguisher up to 7 rounds, and a neural distinguisher up to 5 rounds.

In another study, Bellini et al. [3] introduced an 8-round integral distinguisher and the concept of "weakly-composed-Toffoli gates," which involves specific combinations of Toffoli gates used in the S-box layer of ARADI. This allowed them to extend a given integral distinguisher by one more round without increasing data complexity, demonstrating that the cube-sum in a specific cube is always the same across two words. They also proposed a key recovery attack on 10-round ARADI with a data complexity of 2^{124} and a time complexity of 2^{177}.

On the other hand, [3] experimentally demonstrated that the cube sum in a specific cube of 5 rounds takes the form of "AABB"(ex. 0x9d9dc5c5), although they did not provide a theoretical proof for this. This can be viewed as a type of extended integral distinguisher, and in this paper, we define this characteristic as the "byte-wise equal property". We theoretically proved this property by modifying the division property algorithm. Additionally, we leveraged the observation that a rotation value of 8 significantly impacts this property, allowing us to extend the byte-wise equal property up to 8 rounds. Table 1 compares the data complexities of the algebraic distinguishers for ARADI presented to date.

Round	Data($\log_2(\cdot)$ scale)		
	Section 4	[3]	[1]
5	13	16	84
6	77	84	113
7	112	113	124
8	124	124	-

1.1 Our Contribution

We theoretically prove the experimental 5-round distinguisher with byte-wise equal cube sums presented in [3] by modifying the MILP based division property algorithm to extend the constraints by one round. We observe that this distinguisher arises due to the rotation amount of 8 and extend the byte-wise equal property up to 8 rounds. As a result, we were able to find algebraic distinguishers for rounds 6 and 7 with data complexities of 2^{77} and 2^{112}, respectively, which are lower than those previously reported.

1.2 Organization

Section 2 explains the background knowledge and the structure and key schedule of ARADI. Section 3 describes the extended integral distinguishers and the division property. In Section 4, we prove the byte-wise equal property of 5-round ARADI and extend the property to rounds 6-8. Finally, Section 5 concludes the paper.

2 Preliminary

This chapter provides a brief explanation of algebraic attacks, along with the structure and notation of the ARADI block cipher.

2.1 Boolean Functions and algebraic attack

A Boolean function f defined on n variables is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. This function can be expressed as a polynomial on n variables over $\mathbb{F}_2$, called algebraic normal form (ANF). The algebraic degree of f, denoted $deg(f)$, is defined as the degree of its ANF.

Consider $k = (k_1, ..., k_n)$ as a vector of n secret variables and $v = (v_1, ..., v_m)$ as a vector of m public variables. In this context, each bit of the symmetric-key cryptosystem can be represented by the Boolean function $f(k, v)$. Here, k represents the master key of the block cipher, v represents the plaintext, and $f(k, v)$ denotes the each bit of the ciphertext.

Let a set of public variables $I = \{v_{i_1}, v_{i_1}, ..., v_{i_d}\}$ be a set of cube variables. Then $f(k, v)$ can be rewritten as

$$f(k, v) = t_I \cdot p_I(k, v) \oplus q_I(k, v)$$

where $t_I = \prod_{v \in I} v$, p_I does not include any variables form I, and each term in q_I is not divisible by t_I.

Define C_I, referred to as a cube, as a set of $2^{|I|}$ values where variables in I take all possible combinations of values, and all remaining variables are fixed to some arbitrary values. Then the following equation holds.

$$\bigoplus_{(v_{i_1}, v_{i_1}, ..., v_{i_d}) \in \{0,1\}^d} f(k, v) = p_I(k, v)$$

In this case, if the algebraic degree of f in cube variables is less than d, then f cannot contain a multiple of t_I, which results in $p_I(k, v) = 0$.

2.2 Notations

$\odot$: 32 or 16 bit word-wise AND

$\cdot$: bit-wise AND

$\|$: concatanation of bits array

$S_a^b(x)$: b-bit left rotation of a-bit word x

2.3 Specification of Block Cipher ARADI

The ARADI algorithm uses a block size of 128 bits and a key size of 256 bits, and it consists of 16 rounds.

Round function of ARADI The internal state of ARADI consists of four 32-bit words, and the encryption function is divided into an S-box layer (π), a linear layer (Λ_i), and a key XOR layer (τ_{rk_i}). The overall encryption function is defined as follows:

$$\tau_{rk_{16}} \circ \bigcirc_{i=0}^{15}(\Lambda_{i \bmod 4} \circ \pi \circ \tau_{rk_i})$$

Descriptions of each layer are as follows:

1. **S-box layer(π)**
   It is constructed based on a Toffoli gate operating on a 32-bit words, defined as $(a, b, c) \leftarrow (a, b, c \oplus a \odot b)$

$$X \to X \oplus W \odot Y, Z \to Z \oplus X \odot Y, Y \to Y \oplus W \odot Z, W \to W \oplus X \odot Z$$

2. **linear layer(Λ_i)**
   In round i, the internal state (W, X, Y, Z) is computed as follows.

$$\Lambda_i((W, X, Y, Z)) = (L_i(W), L_i(X), L_i(Y), L_i(Z))$$

   Here, L_i is an involutory linear operation acting on a 32-bit word, which splits a 32-bit input into two 16-bit words (u, l) and performs the following computation.

$$(u, l) \to (u \oplus S_{16}^{a_i}(u) \oplus S_{16}^{c_i}(l), l \oplus S_{16}^{a_i}(l) \oplus S_{16}^{b_i}(u))$$

   The rotation values used in each round i are given in Table .

$i \bmod 4$	a_i	b_i	c_i
0	11	8	14
1	10	9	11
2	9	4	14
3	8	9	7

Table 1. rotation amount

3. **key addition layer(τ_{rk_i})**
   The 128-bit round key rk_i is XORed with the internal state.

Keyschedule of ARADI The key schedule of ARADI has an internal state represented by an array of eight 32-bit words. If the state at step i is denoted as $K_0^i, K_1^i, \ldots, K_7^i$, the 128-bit round key rk^i for the i-th round is defined as $K_0^i \| K_1^i \| K_2^i \| K_3^i$ in even rounds and as $K_4^i \| K_5^i \| K_6^i \| K_7^i$ in odd-indexed rounds.

In each step, $K_0^i \| K_1^i$ and $K_4^i \| K_5^i$ are processed through a 64-bit linear transformation M_0, while $K_2^i \| K_3^i$ and $K_6^i \| K_7^i$ undergo a 64-bit linear transformation

M_1. This is followed by a word-level permutation $P_{i \bmod 2}$, where $P_0 = (1,2)(5,6)$ and $P_1 = (1,4)(3,6)$.

The linear transformations M_0 and M_1 operate on the 64-bit inputs (a, b) as follows:

$$M_0((a,b)) = (S_{32}^1(a) \oplus b, S_{32}^3(b) \oplus S_{32}^1(a) \oplus b)$$
$$M_1((a,b)) = (S_{32}^9(a) \oplus b, S_{32}^{28}(b) \oplus S_{32}^9(a) \oplus b)$$

2.4 Attack Setting for ARADI

We will follow the notations introduced in [3]. Specifically, the state value after r rounds is represented by the 32-bit words W^r, X^r, Y^r, Z^r, and each i-th bit in these words is denoted as $W_i^r, X_i^r, Y_i^r, Z_i^r$. Additionally, the index sets of the bits in the plaintext W^0, X^0, Y^0, Z^0 are represented by $\mathcal{I}_W, \mathcal{I}_X, \mathcal{I}_Y, \mathcal{I}_Z \subseteq \{0, 1, ...31\}$.

When $|\mathcal{I}W| + |\mathcal{I}X| + |\mathcal{I}Y| + |\mathcal{I}Z| = d < 127$, we refer to this as the cube indices of dimension d, or simply as the *cube-index sets*. The corresponding plaintext variables

$$\{W_i^0 | i \in \mathcal{I}_W\} \cup \{X_i^0 | i \in \mathcal{I}_X\} \cup \{Y_i^0 | i \in \mathcal{I}_Y\} \cup \{Z_i^0 | i \in \mathcal{I}_Z\}$$

are used as the cube variables.

3 Extended integral distinguisher and division property

This chapter explains the extended integral distinguisher, the division property, and the method of determining the upper bound of algebraic degree using the MILP-based division property.

3.1 Extended integral distinguisher

Lambin et al. [6] introduced a technique for identifying additional integral distinguishers. Their method focuses on searching for the composition $L_{out} \circ E \circ L_{in}$ rather than directly analyzing a block cipher $E : \mathbb{F}_2^k \times \mathbb{F}_2^n \to \mathbb{F}_2^n$. In this context, L_{in} and L_{out} belong to $GL_n(\mathbb{F}_2)$, and E is considered a nonlinear permutation on $\mathbb{F}_2^n$, functioning as a block cipher with a randomly chosen secret key from $\mathbb{F}_2^k$. Overall, their approach allows for the discovery of an extended integral distinguisher, which is defined as per Definition 1.

Definition 1 ((Extended) Integral Distinguisher). *Let $E : \mathbb{F}_2^k \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be an r-round block cipher with k-bit key and n-bit block. Define $\mathbb{X}$ and $\mathbb{Y}$ as the plaintext and ciphertext multisets of E, respectively. For any key $\kappa \in \mathbb{F}_2^k$, if there exists anon-zero vector $v \in \mathbb{F}_2^n$ such that*

$$\bigoplus_{y \in \mathbb{Y}} v \cdot y = \bigoplus_{x \in \mathbb{X}} v \cdot E_\kappa(x) = 0,$$

then the pair (X, v) is referred to as an r-round integral distinguisher of E, and $v \cdot y$ is called a balanced bit.

In general, searching through all possible L_{in} and L_{out} mappings is infeasible due to the high complexity involved. Therefore, it is essential to consider the structure of the block cipher and focus on specially selected L_{in} and L_{out} mappings for the search.

3.2 Division property

The division property [7] is a technique initially proposed by Todo to find integral distinguishers, which was subsequently extended to include bit-based division properties [8], 3-subset division properties [8], and 3-subset division properties without unknown subsets [5] (3SBDP). In this paper, we utilize 3SBDP modeled through mixed integer linear programming (MILP). The modeling for the basic operations—XOR, AND, COPY, and constant 1 XOR—is as follows.

Basic operations

1. Bitwise XOR : $(b_1, ..., b_n) \rightarrow a : a = b_1 + b_2 + \cdots + b_n$
2. Bitwise AND : $(b_1, ..., b_n) \rightarrow a : a = b_i$ for $\forall i$
3. Bitwise COPY : $(b_1, ..., b_n) \rightarrow a : a \leq b_i$ for $\forall i$ and $b_1 + b_2 + \cdots b_n \leq a$
4. Bitwise XOR with constant 1 : $a \rightarrow b : b \leq a$

We calculated the upper bound on the algebraic degree using the algorithm presented in [3]. Algorithm 1 models the encryption procedure of ARADI, while Algorithm 2 outputs the upper bound of algebraic degree of target bits when considered as a polynomial of cube variables. For a detailed description of the algorithm, please refer to [3].

4 Byte-wise equal property of ARADI

In this chapter, we formally define the byte-wise equal property and prove that 5-round ARADI, with the cube-index set provided in [3], exhibits this property. We then demonstrate that a rotation amount of 8 affects the byte-wise equal property and extend this property up to 8 rounds.

4.1 Experimental 5 round distinguisher on [3]

First, we will provide a precise definition of the byte-wise equal property.

Definition 2 (Byte-wise equal property). *For a 32-bit value (u, l) represented by two 16-bit values, if $S_{16}^8(u) = u, S_{16}^8(l) = l$, we say that (u, l) has the byte-wise equal property. In other words, if the value has the form $AABB$, it is considered to have the byte-wise equal property. Additionally, if the cube sum of a certain cube index set always exhibits the byte-wise equal property, we say that the cube index set has the byte-wise equal property.*

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

input : Empty model $\mathcal{M}$,number of rounds r, key variables $k_0, ..., k_{255}$
output: MILP model $\mathcal{M}$

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

In [3], it was experimentally suggested that using the cube-index set $\mathcal{I}_W = \{11, 12, ..., 23\}, \mathcal{I}_X = \emptyset, \mathcal{I}_Y = \emptyset$ and $\mathcal{I}_Z = \emptyset$, the byte-wise equal property that consecutive bytes in the cube-sum values of X^5 and Z^5 are equal holds for the 5-round ARADI. To prove this, we will first demonstrate that the operation L_i preserves the byte-wise equal property.

Theorem 1. *Assume that a value (u, l) possesses the byte-wise equal property. Then, it follows that $L_i((u, l)) = (U, L)$ also retains the byte-wise equal property.*

Proof. Since (u, l) has the byte-wise equal property, we have $(u, l) = (S_{16}^8(u), S_{16}^8(l))$. Additionally, we can verify through simple calculations that

$$L_i((u, l)) = L_i((S_{16}^8(u), S_{16}^8(l)) = (S_{16}^8(U), S_{16}^8(L))$$

Therefore, it follows that $(U, L) = (S_{16}^8(U), S_{16}^8(L))$ which implies that (U, L) also possesses the byte-wise equal property. ■

According to Theorem 1,

Algorithm 2: [3] MILP model for computing the upper bound on degree

input : Empty model $\mathcal{M}$,number of rounds r, key variables $k_0, ..., k_{255}$, bit position $target$, Indices sets $\mathcal{I}_W, \mathcal{I}_X, \mathcal{I}_Y, \mathcal{I}_Z$
output: Degree upper bound of target function T

Model R rounds ARADI using Algorithm 1
S=$w_0^r \| ... \| w_{31}^r \| x_0^r \| ... \| x_{31}^r \| y_0^r \| ... \| y_{31}^r \| z_0^r \| ... \| z_{31}^r$
for $i = 0 \dots 127$ **do**
  if $i = target$ **then**
    $\mathcal{M}.addConstr(S_i = 1)$
  else
    $\mathcal{M}.addConstr(S_i = 0)$
  $\mathcal{M}.setObjective(\sum_{i \in \mathcal{I}_W} w_i^0 + \sum_{i \in \mathcal{I}_X} X_i^0 + \sum_{i \in \mathcal{I}_y} y_i^0 + \sum_{i \in \mathcal{I}_Z} z_i^0, \text{Maximize})$
return Objective value

$$X^5 = L_0(X^4 \oplus W^4 \odot Y^4), Z^5 = L_0(Z^4 \oplus X^4 \odot Y^4 \oplus W^4 \odot Y^4)$$

having the byte-wise equal property means that

$$L_0(X^5) = X^4 \oplus W^4 \odot Y^4, L_0(Z^5) = Z^4 \oplus X^4 \odot Y^4 \oplus W^4 \odot Y^4$$

also possess the byte-wise equal property.

To demonstrate this, we applied the division property by separating each term. While we could show the byte-wise equal property for X^4, Z^4 and $X^4 \odot Y^4$ by calculating their degrees using Algorithm 1,2, we were unable to establish the byte-wise equal property for $W^4 \odot Y^4$ through Algorithm 1,2.

To enhance the accuracy of the division property, we employed a method that unfolds one round. Specifically, we expanded $W_i^4 \cdot Y_i^4 \oplus W_{i+8}^4 \cdot Y_{i+8}^4$ for $0 \leq i < 8, 16 \leq i < 24$ W^3, X^3, Y^3 and Z^3 into polynomials over W^3, X^3, Y^3 and Z^3 and added them to the MILP model. For example, to find the degree of $W_0^4 \cdot Y_0^4 \oplus W_0^4 \cdot Y_0^4$, we modeled the expanded polynomial in Appendix A as a basic operation and put it into $\mathcal{T}$ of the modified MILP Algorithm 3. This approach allows for the elimination of some monomials during the polynomial expansion process, thereby increasing accuracy of division property. As a result, we verified that all bits are balanced. The specific upper bounds of the algebraic degree are included in Table 2 and Table 3

4.2 Byte-wise equal property for other rounds

Consider the case in the linear layer L_i when $i = 3 \bmod 4$. By dividing the 64-bit value into two 32-bit values, we represent $A = (u_A, l_A), B = (u_B, l_B)$, yielding the following:

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

input : Empty model $\mathcal{M}$,number of rounds r, key variables $k_0, ..., k_{255}$, MILP model $\mathcal{T}$ for target function T

output: MILP model $\mathcal{M}$

$\mathcal{M}.addVar \leftarrow w_i^0, w_i^0, w_i^0, w_i^0$, for $i = 0, ..., 31$
$\mathcal{M}.addVar \leftarrow rk_i^0, a_i^0$, for $i = 0, ..., 127$
RoundkeyXOR($\mathcal{M}, a_0^0, ..., a_{31}^0, w_0^0, ..., w_{31}^0, rk_0^0, ..., rk_{31}^0$)
RoundkeyXOR($\mathcal{M}, a_{32}^0, ..., a_{63}^0, w_{32}^0, ..., w_{63}^0, rk_{32}^0, ..., rk_{63}^0$)
RoundkeyXOR($\mathcal{M}, a_{64}^0, ..., a_{95}^0, w_{64}^0, ..., w_{95}^0, rk_{64}^0, ..., rk_{95}^0$)
RoundkeyXOR($\mathcal{M}, a_{96}^0, ..., a_{127}^0, w_{96}^0, ..., w_{127}^0, rk_{96}^0, ..., rk_{127}^0$)
for $j = 0 \ldots r-2$ **do**
  $\mathcal{M}.addVar \leftarrow b_i^0, c_i^0, d_i^0, e_i^0$, for $i = 0, ..., 31$
  S-box($\mathcal{M}, b_0^0, ..., b_{31}^0, c_0^0, ..., c_{31}^0, d_0^0, ..., d_{31}^0, e_0^0, ..., e_{31}^0$)
  $\mathcal{M}.addVar \leftarrow p_i^j, q_i^j, s_i^j, t_i^{j+1}$, for $i = 0, ..., 31$
  Linear-map($\mathcal{M}, j \bmod 4, p_0^j, ..., p_{31}^j, b_0^j, ..., b_{31}^j$)
  Linear-map($\mathcal{M}, j \bmod 4, q_0^j, ..., q_{31}^j, c_0^j, ..., c_{31}^j$)
  Linear-map($\mathcal{M}, j \bmod 4, s_0^j, ..., s_{31}^j, d_0^j, ..., d_{31}^j$)
  Linear-map($\mathcal{M}, j \bmod 4, t_0^j, ..., t_{31}^j, e_0^j, ..., e_{31}^j$)
  $\mathcal{M}.addVar \leftarrow w_i^{j+1}, x_i^{j+1}, y_i^{j+1}, z_i^{j+1}$, for $i = 0, ..., 31$
  $\mathcal{M}.addVar \leftarrow rk_i^{j+1}$ for $i = 0, ..., 127$
  RoundkeyXOR($\mathcal{M}, w_0^{j+1}, ..., w_{31}^{j+1}, p_0^j, ..., p_{31}^j, rk_0^j, ..., rk_{31}^j$)
  RoundkeyXOR($\mathcal{M}, x_0^{j+1}, ..., x_{31}^{j+1}, q_0^j, ..., q_{31}^j, rk_{32}^j, ..., rk_{63}^j$)
  RoundkeyXOR($\mathcal{M}, y_0^{j+1}, ..., y_{31}^{j+1}, s_0^j, ..., s_{31}^j, rk_{64}^j, ..., rk_{95}^j$)
  RoundkeyXOR($\mathcal{M}, z_0^{j+1}, ..., z_{31}^{j+1}, t_0^j, ..., t_{31}^j, rk_{96}^j, ..., rk_{127}^j$)
Add MILP model $\mathcal{T}$ to $\mathcal{M}$
return MILP model $\mathcal{M}$

$$L_i(A) = L_i((u_A, l_A)) = (u_A \oplus S_{16}^8(u_A) \oplus S_{16}^7(l_A), l_A \oplus S_{16}^8(l_A) \oplus S_{16}^9(u_A))$$

$$L_i(B) = L_i((u_B, l_B)) = (u_B \oplus S_{16}^8(u_B) \oplus S_{16}^7(l_B), l_B \oplus S_{16}^8(l_B) \oplus S_{16}^9(u_B))$$

Since

$$S_{16}^8(u_A \oplus S_{16}^8(u_A)) = u_A \oplus S_{16}^8(u_A),$$
$$S_{16}^8(u_B \oplus S_{16}^8(u_B)) = u_B \oplus S_{16}^8(u_B)$$

we observe that when computing $L_i(A)_j \cdot L_i(B)_j \oplus L_i(A)_{j+8} \cdot L_i(B)_{j+8}$ for $j \in \{0, 1, ..7\}$, 9 terms in total (from 3×3), with 4 of these terms canceling out.

Similarly, since

$$S_{16}^8(l_A \oplus S_{16}^8(l_A)) = l_A \oplus S_{16}^8(l_A)$$
$$S_{16}^8(l_B \oplus S_{16}^8(l_B)) = l_B \oplus S_{16}^8(l_B)$$

when computing $L_i(A)_j \cdot L_i(B)_j \oplus L_i(A)_{j+8} \cdot L_i(B)_{j+8}$ for $j \in \{16, 1, ..23\}$, 9 terms are generated, with 4 of them canceling out.

Therefore we can see that when $i = 3 \bmod 4$, a rotation amount of $a_i = 8$ causes cancellations of monomials in the polynomial when XORing bits that are 8 bits(1 byte) apart. This is the key reason the byte-wise equal property occurs. To leverage this, we added 2 rounds to $i = 3 \bmod 4$ and set $i = 1 \bmod 4$ as the terminal round of the distinguisher for our search. Specifically, we searched so that the distinguishers for rounds 6, 7, and 8 correspond to rounds 4-9, 3-9, and 2-9, respectively.

Since X^{r+1} and Z^{r+1} include X^r, Z^r, $W^r \odot Y^r$ and $X^r \odot Y^r$, we calculated the upper bound of algebraic degree of each component to find cube indices where the upper bound of algebraic degree remains lower than the cube size. The cube indices for $X^r \odot Y^r$ is provided in Table 2, and the cube indices for $W^r \odot Y^r$ is shown in Table 3. For the cubes in Table 2, $X^r \oplus Z^r$ exhibits the byte-wise equal property, and for the cubes in Table 3, both X^r and Z^r display the byte-wise equal property.

The 6-round and 7-round byte-wise equal properties for $X^r \oplus Z^r$ have data complexities of 2^{77} and 2^{112}, respectively. Therefore, we can conclude that they have lower data complexities than the distinguishers presented in [3], which have complexities of 2^{84} and 2^{113}.

5 Conclusion

In this paper, we theoretically prove the byte-wise equal property of 5-round ARADI, as experimentally presented by Bellini et al. in [3]. We utilize the observation that this property is induced by the rotation amount of 8 to extend the byte-wise equal property up to 8 rounds. Consequently, we obtained distinguishers for rounds 6 and 7 with lower data complexities of 2^{77} and 2^{112}, respectively, compared to previous methods. This study suggests that the rotation value of 8, being half of 16, induces undesirable properties. Therefore, it may be beneficial to modify it to other values with similar resistance with other cryptanalysis method.

References

1. 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 (2024)
2. Bellini, E., Gerault, D., Grados, J., Huang, Y.J., Makarim, R., Rachidi, M., Tiwari, S.: Claasp: a cryptographic library for the automated analysis of symmetric primitives. In: International Conference on Selected Areas in Cryptography. pp. 387–408. Springer (2023)

Round r	Indices	Degree of monomial		Cube dimension
		$X_i^r \oplus X_{i+8}^r$	$X_i^r \cdot Y_i^r \oplus$	
		$Z_i^r \oplus Z_{i+8}^r$	$X_{i+8}^r \cdot Y_{i+8}^r$	
4	$\mathcal{I}_W = \{11, ..., 23\}$ $\mathcal{I}_X = \emptyset$ $\mathcal{I}_Y = \emptyset$ $\mathcal{I}_Z = \emptyset$	9	12	13
5	$\mathcal{I}_W = \{0, ..., 19\}$ $\mathcal{I}_X = \{0, ..., 18\}$ $\mathcal{I}_Y = \{0, ..., 18\}$ $\mathcal{I}_Z = \{0, ..., 18\}$	64	76	77
6	$\mathcal{I}_W = \{0, ..., 27\}$ $\mathcal{I}_X = \{0, ..., 27\}$ $\mathcal{I}_Y = \{0, ..., 27\}$ $\mathcal{I}_Z = \{0, ..., 27\}$	104	111	112
7	$\mathcal{I}_W = \{0, ..., 30\}$ $\mathcal{I}_X = \{0, ..., 30\}$ $\mathcal{I}_Y = \{0, ..., 30\}$ $\mathcal{I}_Z = \{0, ..., 30\}$	120	123	124

Table 2. The set of cube indices for $X \odot Y$ and an upper bound on their algebraic degree. The upper bound has been computed for each $i \in \{0, ...7\} \cup \{16, ..., 24\}$ and we represented the maximum value. For each cube indices, $X \oplus Z$ has the byte-wise equal property

Round r	Indices	Degree of monomial		Cube dimension
		$X_i^r \oplus X_{i+8}^r$	$W_i^r \cdot Y_i^r \oplus$	
		$Z_i^r \oplus Z_{i+8}^r$	$W_{i+8}^r \cdot Y_{i+8}^r$	
4	$\mathcal{I}_W = \{11, ..., 23\}$ $\mathcal{I}_X = \emptyset$ $\mathcal{I}_Y = \emptyset$ $\mathcal{I}_Z = \emptyset$	9	12	13
5	$\mathcal{I}_W = \{0, ..., 19\}$ $\mathcal{I}_X = \{0, ..., 19\}$ $\mathcal{I}_Y = \{0, ..., 19\}$ $\mathcal{I}_Z = \{0, ..., 19\}$	67	79	80
6	$\mathcal{I}_W = \{0, ..., 28\}$ $\mathcal{I}_X = \{0, ..., 27\}$ $\mathcal{I}_Y = \{0, ..., 27\}$ $\mathcal{I}_Z = \{0, ..., 27\}$	105	112	113
7	$\mathcal{I}_W = \{0, ..., 30\}$ $\mathcal{I}_X = \{0, ..., 30\}$ $\mathcal{I}_Y = \{0, ..., 30\}$ $\mathcal{I}_Z = \{0, ..., 30\}$	120	123	124

Table 3. The set of cube indices for $W \odot Y$ and an upper bound on their algebraic degree. The upper bound has been computed for each $i \in \{0, ...7\} \cup \{16, ..., 24\}$ and we represented the maximum value. For each cube indices, W and Z has the byte-wise equal property

3. Bellini, E., Rachidi, M., Rohit, R., Tiwari, S.K.: Mind the composition of toffoli gates: Structural algebraic distinguishers of aradi. Cryptology ePrint Archive (2024)
4. Greene, P., Motley, M., Weeks, B.: Aradi and llama: Low-latency cryptography for memory encryption. Cryptology ePrint Archive (2024)
5. 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: 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 39. pp. 466–495. Springer (2020)
6. Lambin, B., Derbez, P., Fouque, P.A.: Linearly equivalent s-boxes and the division property. Designs, Codes and Cryptography **88**, 2207–2231 (2020)
7. Todo, Y.: Structural evaluation by generalized integral property. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 287–314. Springer (2015)
8. Todo, Y., Morii, M.: Bit-based division property and application to simon family. In: Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers 23. pp. 357–377. Springer (2016)

A $W_0^4Y_0^4 + W_8^4Y_8^4$

$$W_0^3W_{23}^3X_0^3Y_0^3Y_{23}^3Z_{23}^3\oplus W_0^3W_{23}^3X_0^3Y_0^3Y_{23}^3\oplus W_0^3W_{23}^3X_0^3Y_0^3\oplus W_0^3W_{23}^3X_{23}^3Y_0^3Y_{23}^3Z_0^3\oplus$$
$$W_0^3W_{23}^3X_{23}^3Y_0^3Y_{23}^3\oplus W_0^3W_{23}^3X_{23}^3Y_{23}^3\oplus W_0^3W_{23}^3Y_0^3Y_{23}^3Z_0^3\oplus W_0^3W_{23}^3Y_0^3Y_{23}^3Z_{23}^3\oplus W_0^3W_{23}^3Y_0^3Z_0^3Z_{23}^3\oplus$$
$$W_0^3W_{23}^3Y_0^3Z_{23}^3 \oplus W_0^3W_{23}^3Y_0^3 \oplus W_0^3W_{23}^3Y_{23}^3Z_0^3Z_{23}^3 \oplus W_0^3W_{23}^3Y_{23}^3Z_0^3 \oplus W_0^3W_{23}^3Y_{23}^3 \oplus$$
$$W_0^3W_{23}^3Z_0^3\oplus W_0^3W_{23}^3Z_{23}^3\oplus W_0^3W_{31}^3X_0^3Y_0^3Y_{31}^3Z_{31}^3\oplus W_0^3W_{31}^3X_0^3Y_0^3Y_{31}^3\oplus W_0^3W_{31}^3X_0^3Y_0^3\oplus$$
$$W_0^3W_{31}^3X_{31}^3Y_0^3Y_{31}^3Z_0^3 \oplus W_0^3W_{31}^3X_{31}^3Y_0^3Y_{31}^3 \oplus W_0^3W_{31}^3X_{31}^3Y_{31}^3 \oplus W_0^3W_{31}^3Y_0^3Y_{31}^3Z_0^3 \oplus$$
$$W_0^3W_{31}^3Y_0^3Y_{31}^3Z_{31}^3\oplus W_0^3W_{31}^3Y_0^3Z_0^3Z_{31}^3\oplus W_0^3W_{31}^3Y_0^3Z_{31}^3\oplus W_0^3W_{31}^3Y_0^3\oplus W_0^3W_{31}^3Y_{31}^3Z_0^3Z_{31}^3\oplus$$
$$W_0^3W_{31}^3Y_{31}^3Z_0^3\oplus W_0^3W_{31}^3Y_{31}^3\oplus W_0^3W_{31}^3Z_0^3\oplus W_0^3W_{31}^3Z_{31}^3\oplus W_0^3X_0^3X_{23}^3Y_0^3Y_{23}^3\oplus W_0^3X_0^3X_{23}^3Y_0^3Z_{23}^3\oplus$$
$$W_0^3X_0^3X_{31}^3Y_0^3Y_{31}^3\oplus W_0^3X_0^3X_{31}^3Y_0^3Z_{31}^3\oplus W_0^3X_{23}^3Y_0^3Y_{23}^3\oplus W_0^3X_{23}^3Y_0^3Z_{23}^3\oplus W_0^3X_{23}^3Y_{23}^3Z_0^3\oplus$$
$$W_0^3X_{23}^3Z_0^3Z_{23}^3 \oplus W_0^3X_{31}^3Y_0^3Y_{31}^3 \oplus W_0^3X_{31}^3Y_0^3Z_{31}^3 \oplus W_0^3X_{31}^3Y_{31}^3Z_0^3 \oplus W_0^3X_{31}^3Z_0^3Z_{31}^3 \oplus$$
$$W_0^3Y_0^3Y_{23}^3Z_0^3\oplus W_0^3Y_0^3Y_{23}^3\oplus W_0^3Y_0^3Y_{31}^3Z_0^3\oplus W_0^3Y_0^3Y_{31}^3\oplus W_0^3Y_{23}^3\oplus W_0^3Y_{31}^3\oplus W_8^3W_{23}^3X_8^3Y_8^3Y_{23}^3Z_{23}^3\oplus$$
$$W_8^3W_{23}^3X_8^3Y_8^3Y_{23}^3 \oplus W_8^3W_{23}^3X_8^3Y_8^3 \oplus W_8^3W_{23}^3X_{23}^3Y_8^3Y_{23}^3Z_8^3 \oplus W_8^3W_{23}^3X_{23}^3Y_8^3Y_{23}^3 \oplus$$
$$W_8^3W_{23}^3X_{23}^3Y_{23}^3\oplus W_8^3W_{23}^3Y_8^3Y_{23}^3Z_8^3\oplus W_8^3W_{23}^3Y_8^3Y_{23}^3Z_{23}^3\oplus W_8^3W_{23}^3Y_8^3Z_8^3Z_{23}^3\oplus W_8^3W_{23}^3Y_8^3Z_{23}^3\oplus$$
$$W_8^3W_{23}^3Y_8^3\oplus W_8^3W_{23}^3Y_{23}^3Z_8^3Z_{23}^3\oplus W_8^3W_{23}^3Y_{23}^3Z_8^3\oplus W_8^3W_{23}^3Y_{23}^3\oplus W_8^3W_{23}^3Z_8^3\oplus W_8^3W_{23}^3Z_{23}^3\oplus$$
$$W_8^3W_{31}^3X_8^3Y_8^3Y_{31}^3Z_{31}^3\oplus W_8^3W_{31}^3X_8^3Y_8^3Y_{31}^3\oplus W_8^3W_{31}^3X_8^3Y_8^3\oplus W_8^3W_{31}^3X_{31}^3Y_8^3Y_{31}^3Z_8^3\oplus$$
$$W_8^3W_{31}^3X_{31}^3Y_8^3Y_{31}^3\oplus W_8^3W_{31}^3X_{31}^3Y_{31}^3\oplus W_8^3W_{31}^3Y_8^3Y_{31}^3Z_8^3\oplus W_8^3W_{31}^3Y_8^3Y_{31}^3Z_{31}^3\oplus W_8^3W_{31}^3Y_8^3Z_8^3Z_{31}^3\oplus$$
$$W_8^3W_{31}^3Y_8^3Z_{31}^3 \oplus W_8^3W_{31}^3Y_8^3 \oplus W_8^3W_{31}^3Y_{31}^3Z_8^3Z_{31}^3 \oplus W_8^3W_{31}^3Y_{31}^3Z_8^3 \oplus W_8^3W_{31}^3Y_{31}^3 \oplus$$
$$W_8^3W_{31}^3Z_8^3\oplus W_8^3W_{31}^3Z_{31}^3\oplus W_8^3X_8^3X_{23}^3Y_8^3Y_{23}^3\oplus W_8^3X_8^3X_{23}^3Y_8^3Z_{23}^3\oplus W_8^3X_8^3X_{31}^3Y_8^3Y_{31}^3\oplus$$
$$W_8^3X_8^3X_{31}^3Y_8^3Z_{31}^3\oplus W_8^3X_{23}^3Y_8^3Y_{23}^3\oplus W_8^3X_{23}^3Y_8^3Z_{23}^3\oplus W_8^3X_{23}^3Y_{23}^3Z_8^3\oplus W_8^3X_{23}^3Z_8^3Z_{23}^3\oplus$$
$$W_8^3X_{31}^3Y_8^3Y_{31}^3 \oplus W_8^3X_{31}^3Y_8^3Z_{31}^3 \oplus W_8^3X_{31}^3Y_{31}^3Z_8^3 \oplus W_8^3X_{31}^3Z_8^3Z_{31}^3 \oplus W_8^3Y_8^3Y_{23}^3Z_8^3 \oplus$$
$$W_8^3Y_8^3Y_{23}^3 \oplus W_8^3Y_8^3Y_{31}^3Z_8^3 \oplus W_8^3Y_8^3Y_{31}^3 \oplus W_8^3Y_{23}^3 \oplus W_8^3Y_{31}^3 \oplus W_{23}^3X_0^3X_{23}^3Y_0^3Y_{23}^3 \oplus$$
$$W_{23}^3X_0^3X_{23}^3Y_{23}^3Z_0^3\oplus W_{23}^3X_0^3Y_0^3Y_{23}^3\oplus W_{23}^3X_0^3Y_0^3Z_{23}^3\oplus W_{23}^3X_0^3Y_{23}^3Z_0^3\oplus W_{23}^3X_0^3Z_0^3Z_{23}^3\oplus$$
$$W_{23}^3X_8^3X_{23}^3Y_8^3Y_{23}^3\oplus W_{23}^3X_8^3X_{23}^3Y_{23}^3Z_8^3\oplus W_{23}^3X_8^3Y_8^3Y_{23}^3\oplus W_{23}^3X_8^3Y_8^3Z_{23}^3\oplus W_{23}^3X_8^3Y_{23}^3Z_8^3\oplus$$
$$W_{23}^3X_8^3Z_8^3Z_{23}^3\oplus W_{23}^3X_{23}^3Z_{23}^3\oplus W_{23}^3Y_0^3Y_{23}^3Z_{23}^3\oplus W_{23}^3Y_0^3Y_{23}^3\oplus W_{23}^3Y_0^3\oplus W_{23}^3Y_8^3Y_{23}^3Z_{23}^3\oplus$$
$$W_{23}^3Y_8^3Y_{23}^3\oplus W_{23}^3Y_8^3\oplus W_{23}^3Z_{23}^3\oplus W_{31}^3X_0^3X_{31}^3Y_0^3Y_{31}^3\oplus W_{31}^3X_0^3X_{31}^3Y_{31}^3Z_0^3\oplus W_{31}^3X_0^3Y_0^3Y_{31}^3\oplus$$
$$W_{31}^3X_0^3Y_0^3Z_{31}^3\oplus W_{31}^3X_0^3Y_{31}^3Z_0^3\oplus W_{31}^3X_0^3Z_0^3Z_{31}^3\oplus W_{31}^3X_8^3X_{31}^3Y_8^3Y_{31}^3\oplus W_{31}^3X_8^3X_{31}^3Y_{31}^3Z_8^3\oplus$$

$$W_{31}^3 X_8^3 Y_8^3 Y_{31}^3 \oplus W_{31}^3 X_8^3 Y_8^3 Z_{31}^3 \oplus W_{31}^3 X_8^3 Y_{31}^3 Z_8^3 \oplus W_{31}^3 X_8^3 Z_8^3 Z_{31}^3 \oplus W_{31}^3 X_{31}^3 Z_{31}^3 \oplus W_{31}^3 Y_0^3 Y_{31}^3 Z_{31}^3 \oplus W_{31}^3 Y_0^3 Y_{31}^3 \oplus W_{31}^3 Y_0^3 \oplus W_{31}^3 Y_8^3 Y_{31}^3 Z_{31}^3 \oplus W_{31}^3 Y_8^3 Y_{31}^3 \oplus W_{31}^3 Y_8^3 \oplus W_{31}^3 Z_{31}^3 \oplus X_0^3 Y_0^3 Y_{23}^3 \oplus X_0^3 Y_0^3 Y_{31}^3 \oplus X_0^3 Y_{23}^3 Z_0^3 \oplus X_0^3 Y_{31}^3 Z_0^3 \oplus X_8^3 Y_8^3 Y_{23}^3 \oplus X_8^3 Y_8^3 Y_{31}^3 \oplus X_8^3 Y_{23}^3 Z_8^3 \oplus X_8^3 Y_{31}^3 Z_8^3 \oplus X_{23}^3 Y_0^3 Y_{23}^3 \oplus X_{23}^3 Y_0^3 Z_{23}^3 \oplus X_{23}^3 Y_8^3 Y_{23}^3 \oplus X_{23}^3 Y_8^3 Z_{23}^3 \oplus X_{23}^3 Y_{23}^3 Z_{23}^3 \oplus X_{23}^3 Y_{23}^3 \oplus X_{31}^3 Y_0^3 Y_{31}^3 \oplus X_{31}^3 Y_0^3 Z_{31}^3 \oplus X_{31}^3 Y_8^3 Y_{31}^3 \oplus X_{31}^3 Y_8^3 Z_{31}^3 \oplus X_{31}^3 Y_{31}^3 Z_{31}^3 \oplus X_{31}^3 Y_{31}^3$$