Improved Differential Cryptanalysis of ARADI

Surajit Mandal[1], Sandip Kumar Mondal[2], Raghvendra Rohit[3] and Santanu Sarkar[1]

[1] Indian Institute of Technology Madras, Chennai
[2] SRM University AP, Andhra Pradesh
[3] Indian Institute of Technology Roorkee, Roorkee
surajit1810@gmail.com,sandipkumarmondal80@gmail.com,raghvendra.rohit@cs.iitr.ac.in,sarkar.santanu.bir@gmail.com

Abstract. This study investigates and improves the differential cryptanalysis of the ARADI block cipher, a low-latency cryptographic system developed by the United States National Security Agency for memory encryption. The preliminary security assessment of ARADI revealed deficiencies that require additional examination. This study revisits and corrects the key recovery attack introduced by Bellini et al. (Indocrypt 2024). By correcting these mistakes, the research presents a precise 11-round key recovery method with revised attack complexities. Additionally, the analysis is expanded to introduce a key recovery attack for 12 rounds of ARADI, leaving the security margin of the cipher to only 4 rounds.

Keywords: ARADI, low-latency block cipher, differential cryptanalysis, key recovery attack

1 Introduction

In the rapidly changing field of cryptography, block ciphers are indispensable for protecting confidential information, especially in memory encryption applications, where low latency is decisive. Memory encryption is pivotal to securing data as it moves between the processor and memory, preserving confidentiality and integrity against threats to exploit vulnerable memory areas.

Low-latency block ciphers have garnered considerable attention, particularly in memory encryption systems. Substantial designs have developed over time, concentrating on delivering superior performance while maintaining robust cryptographic security. Among these, the early PRINCE [13] stands out as a proficient design, later enhanced by versions such as PRINCEv2 [14]. Other designs include SPEEDY [22], QARMA [2], QARMAv2 [3], MANTIS [5], SCARF [15], ORTHROS [4], BipBip [6], Sonic and SuperSonic [7], Gleeok [1], Twinkle [23] etc.

In August 2024, US National Security Agency (NSA) introduced a low-latency block cipher named ARADI [18] along with its associated authentication encryption mode, LLAMA. ARADI features a block size of 128 and a key size of 256 bits. ARADI is specially designed for memory encryption systems. In the design document, the authors did not explicitly express the reason behind the ARADI design; they provided a minimal security assessment. The limited scope of this initial security evaluation has raised concerns regarding the application of various cryptanalytic methods to ensure its security. This gap in the analysis calls for further investigation to assess and establish the security guarantees of ARADI.

1.1 Related Work

In a recent Indocrypt paper, Bellini et al. [8], using the automated tool CLAASP [9], discussed the implementation of ARADI, investigated the randomness behavior of ARADI, the differential and linear trails, and algebraic properties, and in the last section, they gave a key recovery attack on 11 rounds ARADI. Furthermore, in [10, 20], the authors presented algebraic distinguishers for round-reduced ARADI. Also, using the 8-round algebraic distinguisher, a key-recovery attack on 10 rounds was given in [10]. Very recently, Dunkelman and Ghosh [16] but it is algebraic in nature, extended the algebraic attack of [10, 11] and presented key recovery attacks up to 12 rounds of ARADI. In another work, Ghosh et al. [17] presented a 14-round differential meet-in-the-middle attack.

Hu and Jiao [19] analyzed the related key impossible boomerang attack while in [21] the authors evaluated the side-channel security of ARADI.

1.2 Our Contributions

In this paper, we explore the differential cryptanalysis of ARADI. We revisit the key recovery attack of [8] and identify crucial errors in their key-recovery analysis. We then give a refined 11-round key recovery attack on ARADI. By analyzing a toy version of ARADI, we give experimental evidences that verify the correctness of our 11-round attack. We then investigate the challenges while extending the 11-round attack to 12 rounds and consequently provide the theoretical solution for 12 rounds in Theorem 1. In Table 1, we give our attack complexities.

Table 1: Complexity of differential key recovery attacks on ARADI

Rounds	Data	Time	Memory	Source
11	2^{127}	2^{234}	Negligible	Section 3
12	2^{127}	2^{254}	2^{93} bits	Section 4

Note that the presented work is the first differential attack on ARADI in literature that reaches 12 rounds. There exists a 12-round attack by Dunkelman and Ghosh [16] but it is algebraic in nature. The work by Ghosh et al. [17] appeared after the submission of our work to a journal (on 21st August 2025).

1.3 Outline of the Paper

Section 2 introduces the necessary notation and describes the specification of ARADI along with a basic description of differential cryptanalysis. In Section 3, firstly, we mention the error in Bellini et al.'s work and later give the 11 rounds revised key recovery attack. We present the key recovery attack on 12 rounds ARADI in Section 4. In Section 5, we present our analysis on the toy version of ARADI. Finally, Section 6 concludes the paper.

2 Preliminaries

In this section, we first describe the notation used throughout the paper and then give a brief specification of the ARADI block cipher. Next, we review the basics of differential cryptanalysis.

2.1 Notations

The following notations will be used consistently throughout the paper:

- $\oplus$: Bitwise exclusive OR (XOR) operation. (Example: let $a = 010$ and $b = 110$, then $a \oplus b = 100$)

- $\|$: Concatenation. (Example: let $a = 111$ and $b = 000$, then $a \| b = 111000$)
- $\&$: Bitwise AND operation. (Example: let $a = 010$ and $b = 110$, then $a \ \& \ b = 010$)
- $\mathbb{F}_2 = \{0, 1\}$: Finite field consisting of two elements 0 and 1.
- $\mathbb{F}_2^n$: n-dimensional vector space over $\mathbb{F}_2$.
- Enc_K^i : Encryption of i-round ARADI with the secret key K.

2.2 Specification of ARADI

ARADI is a low latency block cipher introduced by Greene et al. [18]. The key size and the block size of ARADI are 256 and 128 bits, respectively. The cipher consists of a total of 16 rounds. The cipher round function is an SPN (substitution permutation network) design and consists of three operations: the S-box layer η, the i-th linear map L_i, and the addition of i-th round key k_i. The expression of the cipher is given by

$$k_{16} \circ \bigcirc_{i=0}^{15}(L_{(i \bmod 4)} \circ \eta \circ k_i)$$

where the composition runs from right to left. Note that the indices of L_i are reduced modulo 4. We now explain the individual operations in detail. In the following, we denote the state of cipher by $W \| X \| Y \| Z$ where W, X, Y, Z are 32-bit words.

Key addition layer The size of round keys is 128 bits, which are generated by the key scheduling algorithm. This operation XORs a 128-bit round key to the 128-bit state. The key scheduling algorithm is completely linear and is described in detail at the end of this subsection.

S-box layer The S-box layer is based on the Toffoli gate applied to three 32-bit words, i.e., $(p, q, r) \rightarrow (p, q, r \oplus (p \ \& \ q))$. So, the η operation transforms the input state (W, X, Y, Z) as follows:

$$\begin{aligned} X &\leftarrow X \oplus (W \ \& \ Y) \\ Z &\leftarrow Z \oplus (X \ \& \ Y) \\ Y &\leftarrow Y \oplus (Z \ \& \ W) \\ W &\leftarrow W \oplus (X \ \& \ Z) \end{aligned}$$

Also, we can represent the S-box layer, η, as 32 identical 4-bit S-boxes (see Table 2). Notice that the input of S-boxes is (w_i, x_i, y_i, z_i) $(i = 0, 1, 2, \ldots, 31)$, i.e., each column follows the above operations. In addition, the 4-bit S-box is given in Table 3. These representations will help better understand the attack.

Table 2: Representation of S-box layer as 32 4-bit S-boxes

W	$\rightarrow$	w_0	w_1	w_2	$\ldots$	w_{31}
X	$\rightarrow$	x_0	x_1	x_2	$\ldots$	x_{31}
Y	$\rightarrow$	y_0	y_1	y_2	$\ldots$	y_{31}
Z	$\rightarrow$	z_0	z_1	z_2	$\ldots$	z_{31}

Table 3: 4-bit S-box of ARADI in integer representation

Input	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Output	0	1	2	3	4	13	15	6	8	11	5	14	12	7	10	9

Linear layer The i-th linear map, L_i is defined as

$$L_i(W, X, Y, Z) \rightarrow (l_i(W), l_i(X), l_i(Y), l_i(Z))$$

where l_i is an involutory linear map on 32-bit words. Let the input of l_i be $a \parallel b$ where a and b are 16-bit words, then the function l_i is defined as follows:

$$l_i(a, b) \rightarrow (a \oplus S_{p_i}(a) \oplus S_{r_i}(b), b \oplus S_{p_i}(b) \oplus S_{q_i}(a))$$

where the operation $S_c(.)$ denotes the left circular shift of a 16-bit word by c positions. The shift values for a round are given in Table 4.

Table 4: Shift amounts in linear map for the i-th round

i mod 4	p_i	q_i	r_i
0	11	8	4
1	10	9	11
2	9	4	14
3	8	9	7

From Table 4, it is easy to see that for each $j = 0, 1, 2, 3$, $\{L_j, L_{j+4}, L_{j+8}, L_{j+12}\}$ are the same map. For example, suppose that we are operating L_{10} which is the same as $L_{10 \bmod 4} = L_2$. But we will use L_{10} to understand the round number better.

Key Schedule Algorithm (KSA) The ARADI's key schedule takes 256-bit secret key and produces 128-bit round keys using two invertible linear maps M_0, M_1, and block-wise permutations. In the following, we describe the entire key schedule procedure (see Figure 1).

The key schedule update varies over a period of two. Two steps (beginning with an even index round) are depicted in Figure 1. Denote the key values at step i by $(K_i^0, K_i^1, K_i^2, K_i^3, K_i^4, K_i^5, K_i^6, K_i^7)$ where each K_i^j is 32-bit. Then for an even index round i, the round key is $K_i^0 \parallel K_i^1 \parallel K_i^2 \parallel K_i^3$ while it is $K_i^4 \parallel K_i^5 \parallel K_i^6 \parallel K_i^7$ for odd i. The linear maps M_0 and M_1 operate on pairs of 32-bit words and are defined as

$$M_0(p, q) = (S_1(p) \oplus q, S_3(q) \oplus S_1(p) \oplus q)$$
$$M_1(p, q) = (S_9(p) \oplus q, S_{28}(q) \oplus S_9(p) \oplus q)$$

where p and q are 32-bit each. The four pairs of consecutive words are combined at each step by applying either M_0 or M_1. Subsequently, a permutation P_k is applied (k is the round modulo 2) and $P_0 = (12)(56)$, $P_1 = (14)(36)$. A counter is XORed into K^7 at each step.

In our paper, we mainly use the round keys k_{10}, k_{11} and k_{12}, where $k_{10} = K_{10}^0 \parallel K_{10}^1 \parallel K_{10}^2 \parallel K_{10}^3$, $k_{11} = K_{11}^4 \parallel K_{11}^5 \parallel K_{11}^6 \parallel K_{11}^7$, $k_{12} = K_{12}^0 \parallel K_{12}^1 \parallel K_{12}^2 \parallel K_{12}^3$.

2.3 Basics of Differential Cryptanalysis

Differential cryptanalysis is a primary method employed in the cryptanalysis of symmetric ciphers. This technique was introduced by Biham and Shamir [12]. Consider a function F that maps from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Let two inputs P_1 and P_2 be processed through the function F and then compute $F(P_1), F(P_2)$. The differential cryptanalysis focuses on the difference (ΔO) between these two computations (where $P_1 \oplus P_2 = \Delta I$ is the input difference and $F(P_1) \oplus F(P_2) = \Delta O$ is the output difference).

A differential in F is a pair $(\Delta I, \Delta O) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ with $\Delta I \neq 0$. Its probability is defined as

$$\Pr_F(\Delta I, \Delta O) = \frac{|\{p \in \mathbb{F}_2^n \mid F(p) \oplus F(p \oplus \Delta I) = \Delta O\}|}{2^n}$$

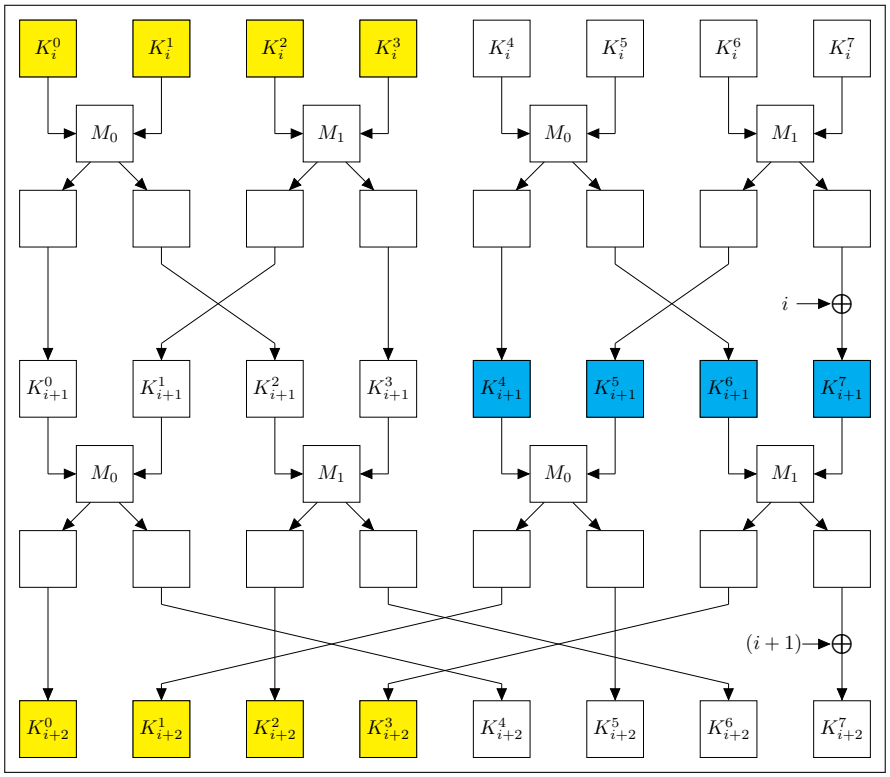


Figure 1: Key Scheduling Algorithm

The Difference Distribution Table (DDT) of F is a $2^n \times 2^n$ array where the $(\Delta I, \Delta O)$ element is given by

$$\mathrm{DDT}[\Delta I, \Delta O] = |\{p \in \mathbb{F}_2^n \mid F(p) \oplus F(p \oplus \Delta I) = \Delta O\}|$$

The DDT of the 4-bit S-box of ARADI is given in Table 5.

Table 5: DDT of ARADI's S-box

ΔI \ ΔO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	0	4	0	0	0	0	0	4	0	4	0	0	0	0
2	0	0	4	0	0	2	2	0	0	0	0	4	0	2	2	0
3	0	0	4	4	0	2	2	0	0	0	0	0	0	2	2	0
4	0	0	0	0	4	2	0	2	0	0	0	0	4	2	0	2
5	0	0	0	0	4	2	0	2	0	0	0	0	4	2	0	2
6	0	0	4	0	0	0	2	2	0	4	0	0	0	0	2	2
7	0	4	4	0	0	0	2	2	0	0	0	0	0	0	2	2
8	**0**	**0**	**0**	**0**	**0**	**2**	**0**	**2**	**4**	**0**	**4**	**0**	**0**	**2**	**0**	**2**
9	0	2	0	2	0	0	4	0	0	2	0	2	4	0	0	0
10	0	2	0	2	2	2	0	0	2	0	2	0	0	0	2	2
11	0	0	0	0	2	0	0	2	2	2	2	2	0	2	2	0
12	0	0	0	0	0	0	4	0	4	0	4	0	4	0	0	0
13	0	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2
14	0	2	0	2	2	0	0	2	2	0	2	0	0	2	2	0
15	0	0	0	0	2	2	0	0	2	2	2	2	0	0	2	2

Now, we know a cipher consists of multiple rounds. Let F represent an r-round iterated cipher expressed as $F = F_{r-1} \circ F_{r-2} \circ \cdots \circ F_0$ where each F_i $(i = 0, 1, \ldots, r-1)$ represents

an individual round. Then, an r-round differential trail is a sequence $(\Delta I_0, \Delta I_1, \ldots, \Delta I_r)$ where each pair $(\Delta I_i, \Delta I_{i+1})$ represents a differential on F_i. Assuming the independence of rounds, the probability of r-round differential trail is defined by

$$\prod_{i=0}^{r-1} \Pr_{F_i}(\Delta I_i, \Delta I_{i+1})$$

where each $\Pr_{F_i}(\Delta I_i, \Delta I_{i+1}) > 0$. In differential analysis, we are interested in trails having a probability significantly greater than 2^{1-n}.

2.4 9-round Differential from [8]

In our analysis, we use the 9-round differential trail from [8]. The trail has a probability 2^{-137} and is given in Table 6. By fixing both the input and output differences of this trail, they computed all possible trails that satisfy this fixed pair of differences. Then, they multiplied all the trail differential probabilities to determine the total probability of the differential. Consequently, the probability of the 9-round differential $(0, 80001002, 0, 0) \longrightarrow (10800010, 0, 0, 0)$ is approximately $2^{-125.5}$.

Table 6: 9-round differential trail

Round	**Difference in word**			
	W	X	Y	Z
0	**0**	**80001002**	**0**	**0**
1	0	00000002	0	0
2	0	10000802	0	0
3	12200811	92200c13	0	0
4	bb365a3f	b9365e3f	0	0
5	d8e0d445	dad1b045	00004000	0
6	30384121	30184120	0	0
7	10100020	00100000	0	0
8	10000000	0	0	0
9	**10800010**	**0**	**0**	**0**

3 Revisiting 11 Rounds Key-recovery Attack on ARADI

In this section, we revisit the key recovery on 11-round ARADI as presented in [8]. First, we explain the error in their work and discuss its implications. Next, we give the correct differential by rectifying this error. In the end, we present the key recovery attack on 11 rounds using the right distinguisher, and discuss how the attack complexities are changed compared to [8].

3.1 Error in Bellini et al.'s Work

The starting point of Bellini et al.'s key recovery procedure is the 9-round differential

$$(0, 80001002, 0, 0) \longrightarrow (10800010, 0, 0, 0)$$

given in Table 6. They extended the output difference $(10800010, 0, 0, 0)$ by two rounds with probability 1. As a result, they claimed that 108 bits become zero, 20 bits are unknown after 10 rounds, and after 11 rounds, 72 bits are zero, and 56 bits are unknown. **However, 36 bits and 88 bits are unknown when we verified this, and the remaining are zeros after 10 and 11 rounds, respectively.**

Table 7: Output difference after 10 rounds, '⋆' represents the difference in that position is unknown, and '0' denotes the difference is zero and $\Delta S_{10} = \Delta W_{10} \parallel \Delta X_{10} \parallel \Delta Y_{10} \parallel \Delta Z_{10}$.

ΔW_{10}	*00*0000**0000*00*00000000**000*
ΔX_{10}	*00*0000**0000*00*00000000**000*
ΔY_{10}	*00*0000**0000*00*00000000**000*
ΔZ_{10}	*00*0000**0000*00*00000000**000*

Table 8: Output difference after 11 rounds, '⋆' represents the difference in that position is unknown, and '0' denotes the difference is zero and $\Delta S_{11} = \Delta W_{11} \parallel \Delta X_{11} \parallel \Delta Y_{11} \parallel \Delta Z_{11}$.

ΔW_{11}	**0*0*0****0****0**0***0*0***00*
ΔX_{11}	**0*0*0****0****0**0***0*0***00*
ΔY_{11}	**0*0*0****0****0**0***0*0***00*
ΔZ_{11}	**0*0*0****0****0**0***0*0***00*

The above error happened due to different linear layers at each round. While extending the differential by two rounds, the correct linear layers are (L_9, L_{10}) and not (L_8, L_9). The later incorrect sequence of linear layers was used in [8].

Consequently, the attack complexities will change due to different number of known and unknown bits at rounds 10 and 11. We rectify this with better improvement in Section 3.2.

We extend the output difference $(\texttt{10800010}, 0, 0, 0)$ another two rounds with probability 1 and give the correct difference in Tables 7 and 8 in matrix form after 10 and 11 rounds, respectively.

3.2 Revised Key Recovery Attack

The key recovery procedure consists of two steps: (1) identify the possible correct pair of plaintexts (P_1, P_2) for which the difference after 9 rounds is $(\texttt{10800010}, 0, 0, 0)$ and (2) recover the secret key using those identified plaintext pairs. In the following, we explain these two steps in detail.

3.2.1 Identifying Possible Correct Pair of Plaintexts

Since the 9-round differential has probability $2^{-125.5}$, we start with 2^{126} pairs of plaintext (P_1, P_2) satisfying $P_1 \oplus P_2 = \Delta S_0 = (0, \texttt{80001002}, 0, 0)$. To identify the pair (P_1, P_2) for which $\Delta S_9 = (\texttt{10800010}, 0, 0, 0)$, we proceed as follows.

1. Call the encryption oracle, i.e., 11-round ARADI for each pair of (P_1, P_2) to have the corresponding (C_1, C_2). In total, we have 2^{126} (C_1, C_2) pairs after 11 rounds.

2. For each (C_1, C_2), compute $C_1 \oplus C_2 = \Delta S_{11}$. Since we know that the 40-bit position difference is zero (see Table 8), we check the difference for all these pairs such that those positions are zero or not. Consequently, the number of pairs will reduce from 2^{126} to $2^{126-40} = 2^{86}$.

   The complexity of this step is 2^{126} 128-bit XORs.

3. After that, for each 2^{86} pairs we move on in backward direction. The difference ΔS_{11} will remain the same after XORing k_{11}. Then, we apply the inverse of linear operation (L_{10}) and get difference $L_{10}^{-1}(\Delta S_{11})$, which is the same as the difference that we obtain after applying the S-box layer on ΔS_{10}. Now, we know that the difference of 36 positions is unknown after ten rounds, and the other $(128 - 36) = 92$ positions is zero (see Table 7). Also, due to the bijective property of the S-box,

input difference zero will always result in zero output difference, and the unknown difference will be unknown.

Therefore, we check whether 92 positions of $L_{10}^{-1}(\Delta S_{11})$ are zero or not. As in the earlier step we filter 2^{40} pairs, in this step we filter $2^{92-40} = 2^{52}$ pairs. Consequently, the number of pairs will reduce from 2^{86} to $2^{86-52} = 2^{34}$.

The complexity of this step is 2^{86} L_{10}^{-1} operations.

4. Now we guess all possible keys in those 36 positions where $L_{10}^{-1}(\Delta S_{11})$ is unknown. As we guess the key, we can apply the inverse S-box to each element of the pairs and get the difference after the inverse S-box. We know the zero difference always goes to the zero difference, so we get the total state difference after the inverse S-box. Now we apply the inverse linear layer, i.e., $L_9^{-1}(\Delta S_{10})$, and check whether this belongs to our required six differences or not (see the set A of Algorithm 1). This is because the input difference 8 can give six nonzero output differences (see Table 5). If it belongs, then the pair is passed.

   The complexity of this step is $2^{34+36} = 2^{70}$ L_9^{-1} operations and $2^{70} \times \frac{9}{32}$ η^{-1} operations.

3.3 Recovering the Key

In this section, we discuss how to recover the key. The core idea is that we move backward two rounds, guess partial round subkeys k_{11} and k_{10}, and verify whether the difference after 9 rounds matches with $\Delta S_9 = (\texttt{10800010}, \texttt{0}, \texttt{0}, \texttt{0})$. The keys k_{11} and k_{10} for which the difference is correct will be considered as the potential candidates for the right keys.

The entire key recovery procedure is depicted in Figure 2 while the individual steps are discussed below.

1. From the key-scheduling algorithm (see Section 2.2), it is evident that if we know the subkeys k_{10} and k_{11}, then we can find the master key simply by inverting the key-scheduling. We obtained 36 linear equations of the key bits k_{11} with $2^{13.86}$ possibilities (See Section 3.5). Now for each possibility we guess specific $(128 - 36) = 92$ bits of k_{11} and find the others bits from these 36 equations (see Appendix A). So, we are left with $2^{92+13.86} = 2^{105.86}$ possible choices of k_{11}.

   Next, for each guess of k_{11}, we do an exhaustive search of k_{10}. We compute the master key K from (k_{11}, k_{10}), and check whether $Enc_K^{11}(P) = C$. Since the block length is 128 bits, we can filter out 2^{128} keys, leading to $2^{105.86+128-128} = 2^{105.86}$ remaining keys.

   The complexity of this step is $2^{105.86+128} = 2^{233.86}$ encryptions.

2. We filter the remaining $2^{105.86}$ keys using another (P', C') pair. We are left with $2^{105.86-128} = 2^{-22.14}$ keys. So, only the correct secret key will survive.

   The complexity of this step is $2^{105.86}$ encryptions.

3.4 Discussion on Attack Complexities

In this section, we present the overall attack complexities of 11-round key recovery attack on ARADI as discussed in previous sections. We also compare them with Bellini et al.'s work.

Data Complexity We require encryption of 2^{126} pairs $(P_1, P_1 \oplus \Delta S_0)$. So, the data complexity is $2 \cdot 2^{126} = 2^{127}$.

Algorithm 1: 11-round key recovery attack on ARADI

Input: 2^{126} plaintext pairs
Output: Secret key K

1 Let $A = \{5, 7, 8, 10, 13, 15\}$
2 $C = Enc_K^{11}(P)$
3 $C' = Enc_K^{11}(P')$
4 $\Delta = (0, 80001002, 0, 0)$
5 **for** $i = 0$ *to* $2^{126} - 1$ **do**
6 Choose $P_1 \in \mathbb{F}_2^{128}$ randomly
7 Set $P_2 = P_1 \oplus \Delta$
8 $C_1 = Enc_K^{11}(P_1)$
9 $C_2 = Enc_K^{11}(P_2)$
10 $\Delta S_{11} = C_1 \oplus C_2$
11 **if** ΔS_{11} *has zero in specific 40 zero positions* **then**
12 $\Delta' = L_{10}^{-1}(\Delta S_{11})$
13 **if** Δ' *has zero in specific 92 zero positions* **then**
14 Guess the 36 values of $L_{10}^{-1}(k_{11})$, other than that specific 92 zero positions of Δ'.
15 Compute η^{-1} partially and get the value of ΔS_{10}.
16 **if** ΔS_{10} *has zero in specific 116 positions and remaining* 3 *columns belongs to the set* A **then**
17 36 key expressions of $L_{10}^{-1}(k_{11})$ is equal to their guessed value.
18 Guess specific 92 bits of subkey k_{11} and find other 36 bits from 36 equations
19 **for** *each* k_{10} **do**
20 Find secret key K from k_{11} and k_{10} using KSA inversion
21 **if** $Enc_K^{11}(P) = C$ **then**
22 **if** $Enc_K^{11}(P') = C'$ **then**
23 K is the secret key
24 **return** K
25 **end**
26 **end**
27 **end**
28 **end**
29 **else**
30 **Goto** line 14 and make a new guess.
31 **end**
32 **end**
33 **end**
34 **end**

Time Complexity Summing the time complexities of each individual step, the final time complexity is given as follows.

$$
\begin{aligned}
& 2^{126} \text{ 128-bit XORs } + 2^{86}\, L_{10}^{-1} + 2^{70}\, L_9^{-1} \text{ operations} \\
& + 2^{70} \times \frac{9}{32}\eta^{-1} \text{ operations} \\
& + (2^{233.86} + 2^{105.86}) \text{ encryptions} \\
& \approx 2^{234} \text{ encryptions}
\end{aligned}
$$

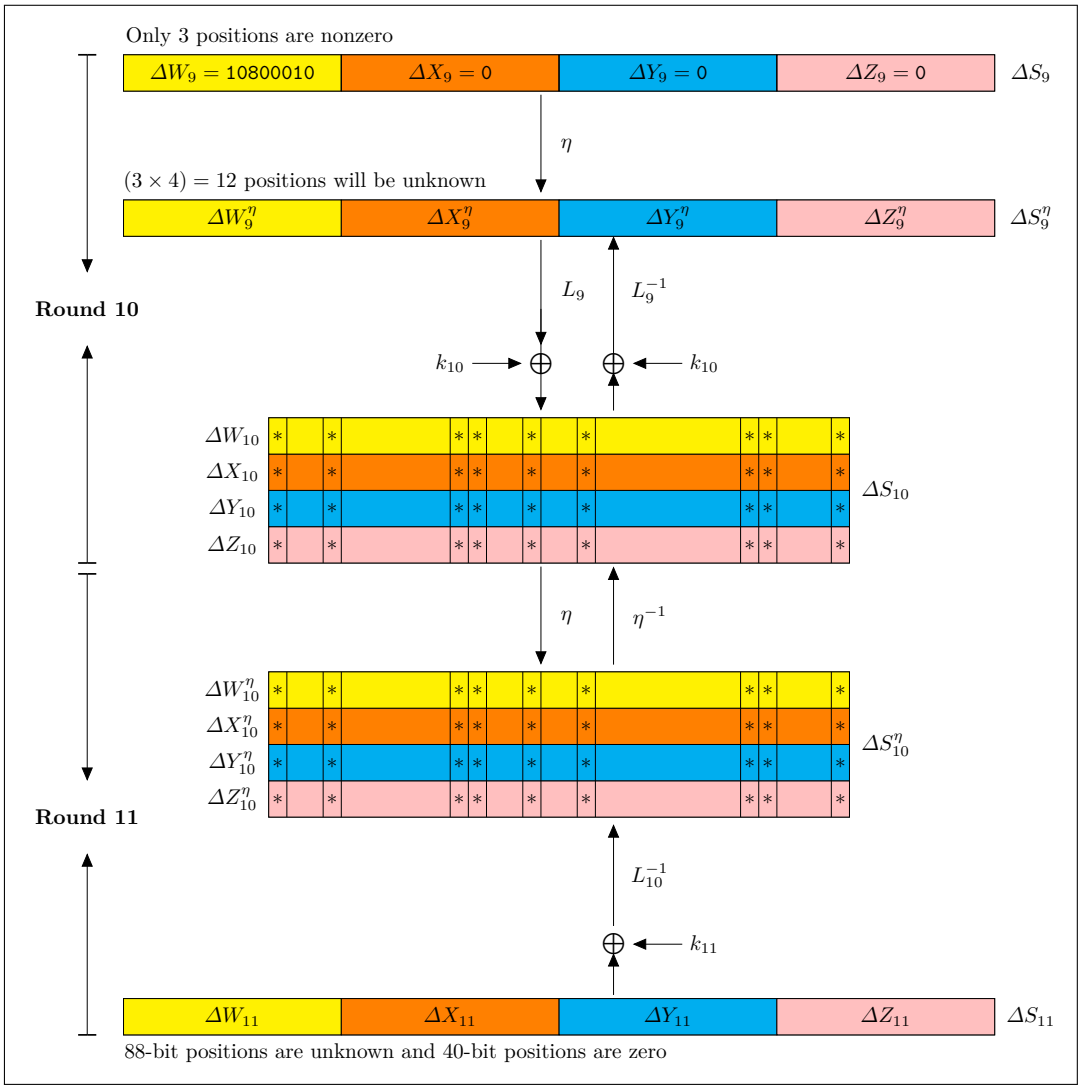


Figure 2: Depiction of 11-round key recovery attack on ARADI. Here '$\star$' denotes bits with unknown difference.

Memory Complexity We do not store any data in both phases of the attack. Hence, the memory complexity is negligible.

3.5 Experimental Results

Due to high computational complexity of the attack, running the entire Algorithm 1 on our machine is infeasible. Therefore, we divide the algorithm into two distinct components: one for processing the 'right pair' and the other for handling the 'wrong pairs'.

For our initial experiment, we select a pair of inputs that exhibit a specific 9-round differential of $(\texttt{10800010}, \texttt{0}, \texttt{0}, \texttt{0})$. This pair was then encrypted through two additional rounds of the ARADI cipher with a fixed secret key. The resulting ciphertexts, after a total of 11 rounds, constitute what we refer to as the right pair. We then determine the number of surviving keys by guessing those 36 positions where $L_{10}^{-1}(\Delta S_{11})$ is unknown, that is, 2^{36} expressions of keys. We filter those keys as described in Algorithm 1.

For our second experiment, we select a random input pair whose differential did not match $(\texttt{10800010}, \texttt{0}, \texttt{0}, \texttt{0})$. We labeled this the wrong pair. As before, we then calculate the number of surviving keys.

After running both experiments 200 times, we found that an average of $14,894$ ($\approx 2^{13.86}$) keys survived for the right pair. Surprisingly, not a single key survived for the wrong pair.

4 Key Recovery Attack on 12 rounds ARADI

The best-known attacks on ARADI reach up to 11 rounds (incorrect attack by Bellini et al. and the rectified one in Section 3). Thus, it is natural to ask whether we can extend the former attacks by 1 more round. A naive strategy would be to extend the 9-round differential to 3 more rounds with probability 1, but this results in all 128-bit positions, i.e., ΔS_{12} becomes unknown. This makes filtering the correct pairs of (P_1, P_2) challenging.

In this section, we study how to solve the above challenges and present the first-ever differential based key-recovery attack on 12 rounds ARADI.

4.1 Identifying Correct Pair of Plaintexts

The identification of a correct pair of plaintexts is explained step by step as follows.

1. We have 2^{126} (C_1, C_2) pairs after 12 rounds. For each pair, calculate $C_1 \oplus C_2 = \Delta S_{12}$. The complexity of this step is 2^{126} 128-bit XORs.

2. After that, for each of 2^{126} pairs, we move on in a backward direction. The difference ΔS_{12} will remains same after XORing k_{12}. Then, we compute $L_{11}^{-1}(\Delta S_{12})$. Now, for a correct pair, 88 position differences should be unknown and 40 positions should be zero after this operation (see Figure 3). Therefore, we check whether 40 positions of $L_{11}^{-1}(\Delta S_{12})$ are zero or not. This will filter 2^{40} pairs reducing the possible pairs from 2^{126} to $2^{(126-40)} = 2^{86}$.

   The complexity of this step is 2^{126} L_{11}^{-1} operations.

3. Now, we guess the values of 88 linear expressions of keys from k_{12}, i.e., the expressions in the 88 position of $L_{11}^{-1}(k_{12})$ (see Appendix B). For each guessed value of 88 linear expressions we fix a counter. Therefore, we have 2^{88} counters and $2^{88+86} = 2^{174}$ pairs.

   Since 88 unknown positions are in 21 columns, we can apply the S-box inverse operations in those 21 columns. Note that other columns have zero difference, so their difference after the S-box inverse remains zero. Thus, we get the difference ΔS_{11}.

   The complexity of this step is $2^{174} \times \frac{21}{32}$ η^{-1} operations.

4. Now, we compute $L_{10}^{-1}(\Delta S_{11})$. Since there are 36 unknowns and 92 zeros after ten rounds ARADI for a right pair, so after S-box operation, there will be 36 unknowns and 92 zeros (see Figure 2). This means in $L_{10}^{-1}(\Delta S_{11})$ there are also 36 unknown and 92 zeros. So we check whether 92 positions of $L_{10}^{-1}(\Delta S_{11})$ are zero or not. Note that we have guessed 88 expressions of key, thus the number of pairs will reduce from 2^{174} to $2^{(174-88)} = 2^{86}$.

   The complexity of this step is 2^{174} L_{10}^{-1} operations.

After Step 4, we have 2^{88} counters and 2^{86} pairs. A counter corresponding to the key guess increases when the ciphertext pair is not eliminated. Accordingly, there could be multiple counters which can increase and thus we can not eliminate 2^{86} pairs in a trivial manner.

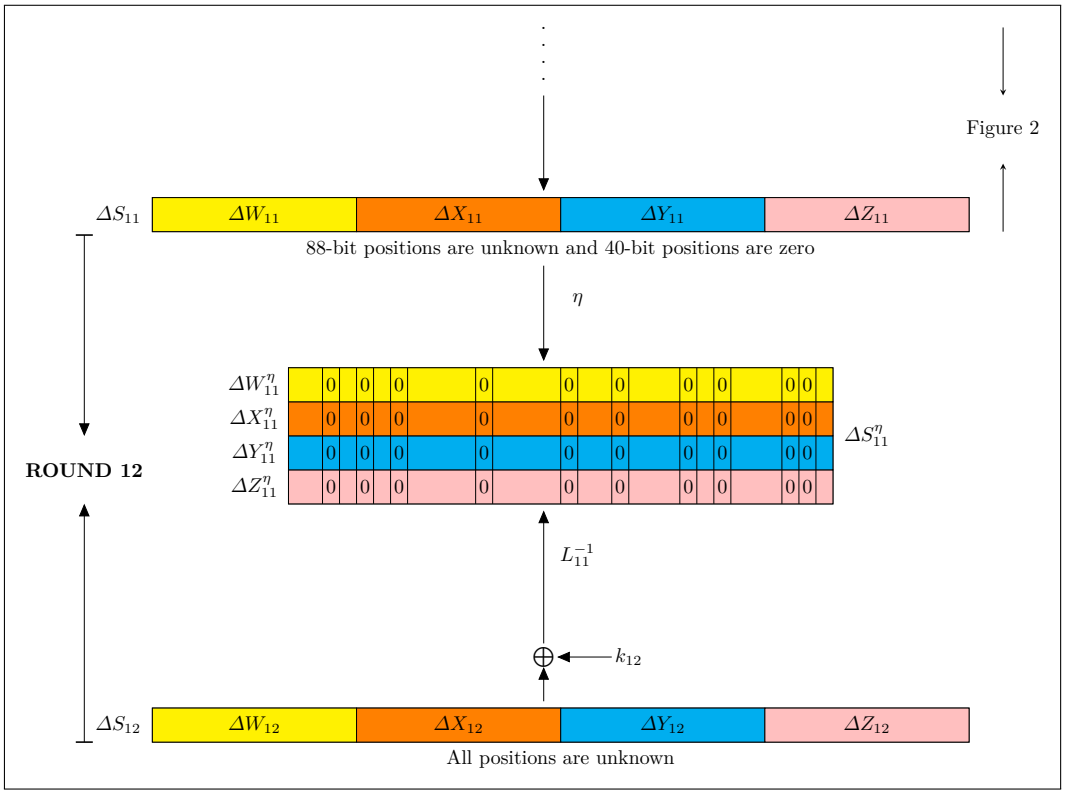


Figure 3: Depiction of 12-round key recovery attack on ARADI

In Theorem 1, we will determine how many counters will increase at least by one.

Theorem 1. *Let there be $N = 2^n$ counters. If we increase those counters randomly one at a time up to P ($P \leq N$) times, then the expected number of counters that increases by at least 1 is $\frac{\beta^P - 1}{\beta - 1}$, where $\beta = 1 - \frac{1}{N}$.*

Proof. There are total N counters. We have to find the expected number of counters that increases by at least 1.

Let E_i be the expected number of counter increases where the total counter increases is i. Therefore, $E_0 = 0, E_1 = 1$. We will find E_i, now E_{i-1} is the expected number of counter increases when total counter increases is $(i-1)$. When we are adding extra one, the probability that extra counter increases outside of E_{i-1} is $\frac{N - E_{i-1}}{N}$. Therefore, $E_i = E_{i-1} + \frac{N - E_{i-1}}{N}$.

Now, to find, E_P we proceed as follows. First, we solve the recurrence relation

$$E_i = E_{i-1} + \frac{N - E_{i-1}}{N} = E_{i-1} + 1 - \frac{E_{i-1}}{N} = E_{i-1}\beta + 1$$

where $\beta = 1 - \frac{1}{N}$. Assume that

$$\begin{aligned} B &= \sum_{i=1}^{\infty} E_i z^i \\ &= \sum_{i=1}^{\infty} (E_{i-1}\beta + 1) z^i \\ &= \sum_{i=1}^{\infty} (E_{i-1}\beta z^i + z^i) \\ &= \beta z \sum_{i=1}^{\infty} E_{i-1} z^{i-1} + \sum_{i=1}^{\infty} z^i \\ &= \beta z B + \frac{z}{1-z} \text{ [as } E_0 = 0] \end{aligned}$$

Therefore, from the above equation we obtain

$$\begin{aligned} B &= \frac{z}{(1-z)(1-\beta z)} \\ &= \frac{1}{\beta - 1} \cdot \frac{\beta z - z}{(1-z)(1-\beta z)} \\ &= \frac{1}{\beta - 1} \left(\frac{1}{1-\beta z} - \frac{1}{1-z} \right) \\ &= \frac{1}{\beta - 1} \left(\sum_{i=0}^{\infty} \beta^i z^i - \sum_{i=0}^{\infty} z^i \right) \\ &= \frac{1}{\beta - 1} \sum_{i=1}^{\infty} (\beta^i - 1) z^i \end{aligned}$$

Hence,

$$\sum_{i=1}^{\infty} E_i z^i = \sum_{i=1}^{\infty} \frac{\beta^i - 1}{\beta - 1} z^i \text{ [since } B = \sum_{i=1}^{\infty} E_i z^i]$$

which implies, $E_i = \frac{\beta^i - 1}{\beta - 1}$. So, $E_P = \frac{\beta^P - 1}{\beta - 1}$.

□

In our case, $N = 2^{88}$, $\beta = 1 - 2^{-88}$, $P = 2^{86}$. Therefore, $E_{2^{86}} = \frac{\beta^{2^{86}} - 1}{\beta - 1} \approx 2^{85.82}$.

Remark 1. In [8], the authors evaluated that the expected number of counters increased by 1 is 2^{72}. If we match their problem with Theorem 1, then $N = 2^{76}, P = 2^{74}$, and we get the answer as $2^{73.82}$.

In the next section, we will explain how the secret key is recovered from the potential key candidates outlined above.

4.2 Key Recovery Procedure

Following Theorem 1, we are left with $2^{85.82}$ possible set of linear equations of key bit from k_{12}. For each possible set we guess $(128 - 88) = 40$ bits of k_{12} such that the others 88 bits can be derived from the linear equations and finally we get complete k_{12}.

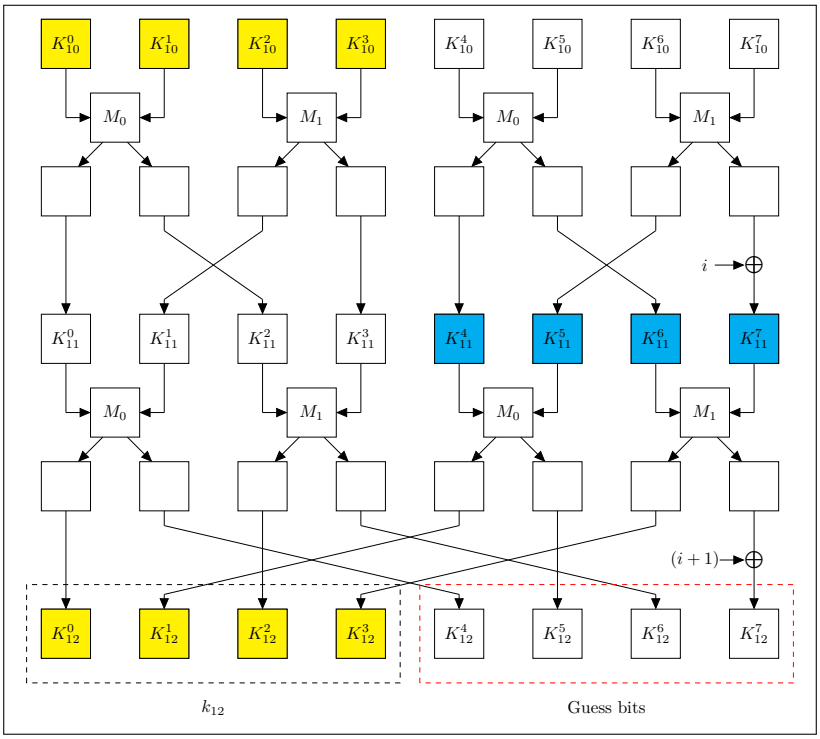


Figure 4: Depiction of guessed key bits for 12-round key recovery attack

Recall that $k_{12} = K_{12}^0 \parallel K_{12}^1 \parallel K_{12}^2 \parallel K_{12}^3$. Now instead of guessing k_{11}, we guess (see Figure 4) $K_{12}^4, K_{12}^5, K_{12}^6$ and K_{12}^7. Knowing $(K_{12}^0, K_{12}^1, K_{12}^2, K_{12}^3, K_{12}^4, K_{12}^5, K_{12}^6, K_{12}^7)$ we can invert the key-scheduling and find the master key K.

In total, we have $2^{85.82+40+128} = 2^{253.82}$ possible choice of K. For each K, we check whether $Enc_K^{12}(P) = C$. Since the block length is 128 bits we can filter out 2^{128} keys, leading to $2^{253.82-128} = 2^{125.82}$ remaining keys. The complexity of this step is $2^{253.82}$ encryptions.

We filter the remaining $2^{125.82}$ keys using another (P', C') pair. We are left with $2^{125.82-128} = 2^{-2.18}$ keys. So, only the correct secret key will survive. The complexity of this step is $2^{125.82}$ encryptions.

The entire procedure is given in Algorithm 2.

4.3 Discussion on Attack Complexities

In this section, we present the overall attack complexities of 12-round key recovery attack on ARADI as discussed in previous section.

Data Complexity We require encryption of 2^{126} pairs $(P_1, P_1 \oplus \Delta S_0)$. So the data complexity is $2 \cdot 2^{126} = 2^{127}$.

Algorithm 2: 12-round key recovery attack

Input: 2^{126} plaintext pairs
Output: Secret Key K

1 $\Delta = (0, 80001002, 0, 0)$
2 **for** $i = 0$ *to* $2^{126} - 1$ **do**
3 Choose $P_1 \in \mathbb{F}_2^{128}$ randomly
4 $P_2 = P_1 \oplus \Delta$
5 $C_1 = Enc_K^{12}(P_1)$
6 $C_2 = Enc_K^{12}(P_2)$
7 $\Delta S_{12} = C_1 \oplus C_2$
8 $\Delta' = L_{11}^{-1}(\Delta S_{12})$
9 **if** Δ' *has zeros in specific 40 positions* **then**
10 Guess the 88 positions of $L_{11}^{-1}(k_{12})$ where $L_{11}^{-1}(\Delta S_{12})$ is unknown and make a counter for every guess.
11 Compute η^{-1} partially and get the value of ΔS_{11}.
12 **if** ΔS_{11} *has zeros in specific 92 positions* **then**
13 Increase the counter by 1
14 **end**
15 **else**
16 **Goto** line 10 and make a new guess
17 **end**
18 **end**
19 **end**
20 If a counter increases then 88 linear expressions of $L_{11}^{-1}(k_{12})$ are equal to the counter value. Find the subkey k_{12} by guessing 40 bits and the rest from the 88 equations.
21 Guess the values of $K_{12}^4, K_{12}^5, K_{12}^6$ and K_{12}^7.
22 Find the secret key K by inverting the key scheduling algorithm.
23 Choose two plaintext ciphertext pairs (P, C) and (P', C').
24 **if** $Enc_K^{12}(P) = C$ **then**
25 **if** $Enc_K^{12}(P') = C'$ **then**
26 K is the secret key
27 **end**
28 **end**
29 **return** *Secret key* K

Time Complexity Summing the time complexities of each individual step, the final time complexity is given as follows.

$$\begin{aligned}
&2^{126} \text{ 128-bit XORs } + 2^{126}\, L_{11}^{-1} + 2^{174}\, L_{10}^{-1} \text{ operations} \\
&+ 2^{175} \times \frac{21}{32}\, \eta^{-1} \text{ operations } + (2^{253.82} + 2^{125.82}) \text{ encryptions} \\
&\approx 2^{254} \text{ encryptions}
\end{aligned}$$

Memory Complexity We need to store 2^{86} counters. Assume the worst case, i.e., each counter is of size equal to the block length which is 128 bits.[1] Then, the memory complexity is $2^{86} \times 128 = 2^{93}$ bits.

5 Attack Analysis on Toy version of ARADI

This section presents an attack on toy version of ARADI to justify the attack on 11-round, given in the previous section. We first give a brief specification of toy ARADI and then explain our attack which is supported by practical experiments.

5.1 Specification of Toy version of ARADI

Let the block size of the toy ARADI cipher as 32 bits, so there are a total of 8 S-boxes, and the S-box is the same as the ARADI cipher. We keep the same notations used in the main cipher for better understanding. Here, the shift amount will be changed; thus, the linear layer is defined as follows.

$$L_i(W, X, Y, Z) \rightarrow (l_i(W), l_i(X), l_i(Y), l_i(Z))$$

where l_i is an involutory linear map on 8-bit words (it depends on the condition $2p_i = q_i + r_i$, so we have chosen in that way). Let the input of l_i be $a \parallel b$ where a and b are 4-bit words, then the function l_i is defined as follows:

$$l_i(a, b) \rightarrow (a \oplus S_{p_i}(a) \oplus S_{r_i}(b), b \oplus S_{p_i}(b) \oplus S_{q_i}(a))$$

where the operation $S_c(.)$ denotes the left circular shift of a 16-bit word by c positions. The shift values for a round is given in Table 9.

Table 9: Shift amounts on Toy version in linear map for the i-th round

i mod 4	p_i	q_i	r_i
0	2	1	3
1	1	0	2
2	0	3	1
3	3	2	0

Key Scheduling Algorithm Analogous to the actual key schedule of ARADI, we have defined a key scheduling algorithm for the toy version. We take 64-bit main key and generate 32-bit round keys using the same block-wise permutations $P_0 = (12)(56)$ and $P_1 = (14)(36)$ and two invertible linear maps M_0, M_1 operate on pairs of 8-bit, which are defined as follows.

$$\begin{aligned}
M_0(p, q) &= (S_1(p) \oplus q, S_3(q) \oplus S_1(p) \oplus q) \\
M_1(p, q) &= (S_3(p) \oplus q, S_6(q) \oplus S_3(p) \oplus q)
\end{aligned}$$

[1] In best case it can be even 2 bits.

5.2 Attack Analysis

We have taken the 9 round differential as $(\mathtt{10}, \mathtt{00}, \mathtt{00}, \mathtt{00})$ and extended it by two rounds with probability 1 (see Figure 5). As a result, 12 bits become unknown, and 20 bits are zero after 10 rounds, and after 11 rounds, 4 bits are zero. Otherwise, all are unknown.

In our experiments, we started by making one right pair and randomly took the wrong pairs. We guess all the possible keys of the 12 expressions of $L_{10}^{-1}(\Delta S_{11})$, where it is unknown. Using the known values from our pairs at these 12 positions, we have applied the inverse S-box to find the resulting difference. Now, we get the complete state difference after the inverse S-box. Next, we have applied $L_9^{-1}(\Delta S_{10})$ and verified whether the difference belongs to our six required differences or not.

Due to the high computational complexity, our analysis on the main ARADI cipher was restricted to testing only a single wrong pair. However, for the toy version of the cipher, we are able to justify all possible wrong pairs. From our code, we have observed that, on average, 24 of the 2^{12} keys are passed for the right pair. Moreover, no key is passed for the wrong pairs.

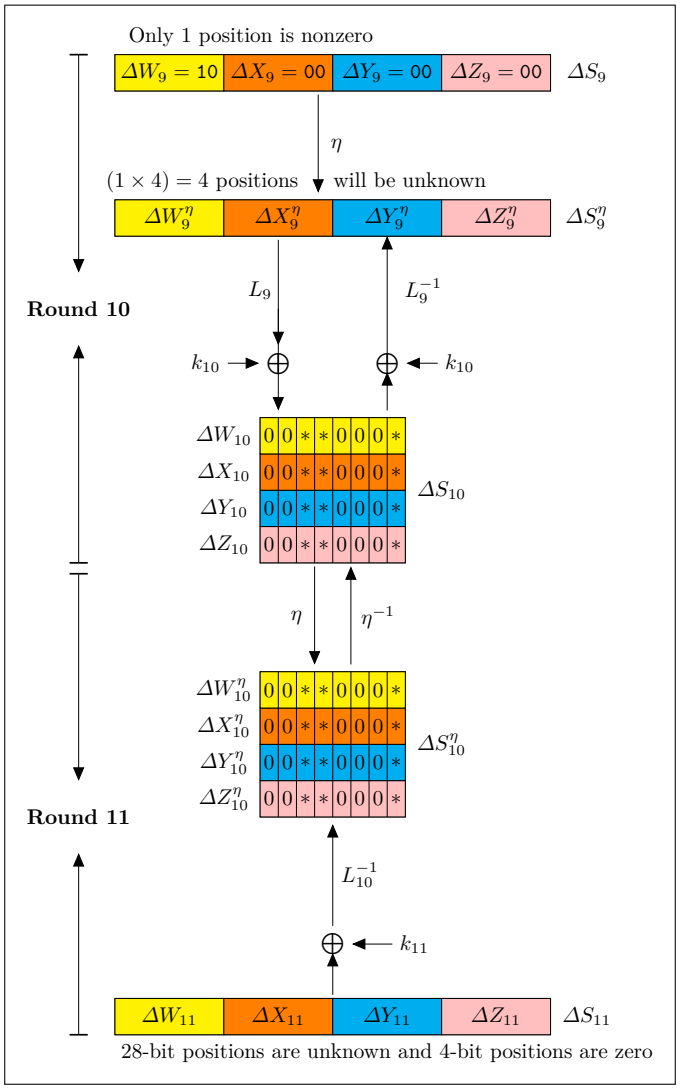


Figure 5: Depiction of 11-round key recovery attack on toy ARADI

5.3 Impact of Toy Version Analysis

From the toy version analysis, we find that no pair is survived for any wrong pair. Also, we have seen that in the main version, for randomly chosen 200 wrong pairs, no key is survived. So, we expect that in the original version, no key will survive for wrong pairs. And for the right pair, some wrong keys also survived. Hence, we give the average number of wrong keys that survive for the right pair.

6 Conclusion

In this work, we have presented the improved differential cryptanalysis on ARADI. At first, we revisit the key recovery attack proposed by Bellini et al. and identify the error in their approach. By addressing those errors, we give a refined 11-round key recovery technique with time and data complexities 2^{234} and 2^{127}, respectively, and the memory complexity is negligible. We then presented the 12-round key recovery attack on ARADI in the literature with time, data and memory complexities of 2^{254}, 2^{127} and 2^{93}, respectively.

As a future work, we aim to improve the complexities of our attacks and the number of attacked rounds by looking in-depth at ARADI's key schedule and finding better differential distinguishers.

References

[1] Ravi Anand, Subhadeep Banik, Andrea Caforio, Tatsuya Ishikawa, Takanori Isobe, Fukang Liu, Kazuhiko Minematsu, Mostafizar Rahman, and Kosei Sakamoto. Gleeok: A family of low-latency prfs and its applications to authenticated encryption. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(2):545–587, 2024.

[2] Roberto Avanzi. 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.

[3] Roberto Avanzi, Subhadeep Banik, Orr Dunkelman, Maria Eichlseder, Shibam Ghosh, Marcel Nageler, and Francesco Regazzoni. The qarmav2 family of tweakable block ciphers. *IACR Trans. Symmetric Cryptol.*, 2023(3):25–73, 2023.

[4] Subhadeep Banik, Takanori Isobe, Fukang Liu, Kazuhiko Minematsu, and Kosei Sakamoto. Orthros: A low-latency PRF. *IACR Trans. Symmetric Cryptol.*, 2021(1):37–77, 2021.

[5] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[6] Yanis Belkheyar, Joan Daemen, Christoph Dobraunig, Santosh Ghosh, and Shahram Rasoolzadeh. Bipbip: A low-latency tweakable block cipher with small dimensions. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):326–368, 2023.

[7] Yanis Belkheyar, Joan Daemen, Christoph Dobraunig, Santosh Ghosh, and Shahram Rasoolzadeh. Introducing two low-latency cipher families: Sonic and supersonic. *IACR Cryptol. ePrint Arch.*, page 878, 2023.

[8] Emanuele Bellini, Mattia Formenti, David Gérault, Juan Grados, Anna Hambitzer, Yun Ju Huang, Paul Huynh, Mohamed Rachidi, Raghvendra Rohit, and Sharwan K. Tiwari. Claasping aradi: Automated analysis of the aradi block cipher. In Sourav Mukhopadhyay and Pantelimon Stănică, editors, *Progress in Cryptology – INDOCRYPT 2024*, pages 90–113, Cham, 2025. Springer Nature Switzerland.

[9] Emanuele Bellini, David Gérault, Juan Grados, Yun Ju Huang, Rusydi H. Makarim, Mohamed Rachidi, and Sharwan K. Tiwari. CLAASP: A cryptographic library for the automated analysis of symmetric primitives. In Claude Carlet, Kalikinkar Mandal, and Vincent Rijmen, editors, *Selected Areas in Cryptography - SAC 2023 - 30th International Conference, Fredericton, Canada, August 14-18, 2023, Revised Selected Papers*, volume 14201 of *Lecture Notes in Computer Science*, pages 387–408. Springer, 2023.

[10] Emanuele Bellini, Mohamed Rachidi, Raghvendra Rohit, and Sharwan K. Tiwari. Mind the composition of toffoli gates: Structural algebraic distinguishers of ARADI. *IACR Cryptol. ePrint Arch.*, page 1559, 2024.

[11] Emanuele Bellini, Mohamed Rachidi, Raghvendra Rohit, and Sharwan K. Tiwari. On the structural properties of toffoli gate composition in ARADI: implications for algebraic distinguishers. In Marc Fischlin and Veelasha Moonsamy, editors, *Applied Cryptography and Network Security - 23rd International Conference, ACNS 2025, Munich, Germany, June 23-26, 2025, Proceedings, Part II*, volume 15826 of *Lecture Notes in Computer Science*, pages 400–425. Springer, 2025.

[12] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[13] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *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*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[14] Dusan Bozilov, Maria Eichlseder, Miroslav Knezevic, Baptiste Lambin, Gregor Leander, Thorben Moos, Ventzislav Nikov, Shahram Rasoolzadeh, Yosuke Todo, and Friedrich Wiemer. Princev2 - more security for (almost) no overhead. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 483–511. Springer, 2020.

[15] Federico Canale, Tim Güneysu, Gregor Leander, Jan Philipp Thoma, Yosuke Todo, and Rei Ueno. SCARF - A low-latency block cipher for secure cache-randomization. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 1937–1954. USENIX Association, 2023.

[16] Orr Dunkelman and Shibam Ghosh. Improved key-recovery attacks on aradi. *Cryptology ePrint Archive*, 2025.

[17] Shibam Ghosh, Bastien Michel, and María Naya-Plasencia. Differential-mitm attack on 14-round aradi. *Cryptology ePrint Archive*, 2025.

[18] Patricia Greene, Mark Motley, and Bryan Weeks. ARADI and LLAMA: low-latency cryptography for memory encryption. *IACR Cryptol. ePrint Arch.*, page 1240, 2024.

[19] Xichao Hu and Lin Jiao. Pre-sieve, partial-guess, and accurate estimation: Full-round related-key impossible boomerang attack on ARADI. Cryptology ePrint Archive, Paper 2025/056, 2025.

[20] Sunyeop Kim, Insung Kim, Dongjae Lee, Deukjo Hong, Jaechul Sung, and Seokhie Hong. Byte-wise equal property of aradi. *Cryptology ePrint Archive*, 2024.

[21] Donggeun Kwon and Seokhie Hong. Side-channel attack on ARADI. Cryptology ePrint Archive, Paper 2024/1987, 2024.

[22] Gregor Leander, Thorben Moos, Amir Moradi, and Shahram Rasoolzadeh. 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.

[23] Jianhua Wang, Tao Huang, Shuang Wu, and Zilong Liu. Twinkle: A family of low-latency schemes for authenticated encryption and pointer authentication. *IACR Commun. Cryptol.*, 1(2):20, 2024.

A Guessed Linear Expressions for 11-round ARADI

Let $W \parallel X \parallel Y \parallel Z$ be the state. Since $l_i(W), l_i(X), l_i(Y), l_i(Z)$ are independent of each other, it is enough to compute the linear expressions corresponding to $l_{10}^{-1}(k_{11}^0 \parallel k_{11}^1 \parallel \ldots \parallel k_{11}^{31})$ where $K_{11}^4 = k_{11}^0 \parallel \ldots \parallel k_{11}^{31}$. The exact linear expressions for the 9 required bit positions are given in Table 10.

Table 10: Linear expressions for the 9 required bit positions

Bit index	**Linear expression**
0	$k_{11}^0 \oplus k_{11}^9 \oplus k_{11}^{30}$
3	$k_{11}^3 \oplus k_{11}^{12} \oplus k_{11}^{17}$
8	$k_{11}^8 \oplus k_{11}^1 \oplus k_{11}^{22}$
9	$k_{11}^9 \oplus k_{11}^2 \oplus k_{11}^{23}$
14	$k_{11}^{14} \oplus k_{11}^7 \oplus k_{11}^{28}$
17	$k_{11}^{17} \oplus k_{11}^{26} \oplus k_{11}^5$
26	$k_{11}^{26} \oplus k_{11}^{19} \oplus k_{11}^{14}$
27	$k_{11}^{27} \oplus k_{11}^{20} \oplus k_{11}^{15}$
31	$k_{11}^{31} \oplus k_{11}^{24} \oplus k_{11}^3$

Similarly, we compute the 9 linear expressions for $l_{10}^{-1}(k_{11}^{32} \parallel k_{11}^{33} \parallel \ldots \parallel k_{11}^{63})$, $l_{10}^{-1}(k_{11}^{64} \parallel k_{11}^{65} \parallel \ldots \parallel k_{11}^{95})$ and $l_{10}^{-1}(k_{11}^{96} \parallel k_{11}^{97} \parallel \ldots \parallel k_{11}^{127})$ resulting in total $9 \times 4 = 36$ linear expressions. Here we have

$$K_{11}^5 = k_{11}^{32} \parallel k_{11}^{33} \parallel \ldots \parallel k_{11}^{63}$$
$$K_{11}^6 = k_{11}^{64} \parallel k_{11}^{65} \parallel \ldots \parallel k_{11}^{95}$$
$$K_{11}^7 = k_{11}^{96} \parallel k_{11}^{97} \parallel \ldots \parallel k_{11}^{127}$$

B Guessed Linear Expressions for 12-round ARADI

In Table 11, we give the 22 linear expressions which are to be guessed after applying $l_{11}^{-1}(k_{12}^0 \parallel k_{12}^1 \parallel \ldots \parallel k_{12}^{31})$ where $K_{12}^0 = k_{12}^0 \parallel \ldots \parallel k_{12}^{31}$.

Table 11: Linear expressions for the 22 required bit positions

Bit index	Linear expression	Bit index	Linear expression
0	$k_{12}^0 \oplus k_{12}^8 \oplus k_{12}^{23}$	15	$k_{12}^{15} \oplus k_{12}^7 \oplus k_{12}^{22}$
1	$k_{12}^1 \oplus k_{12}^9 \oplus k_{12}^{24}$	17	$k_{12}^{17} \oplus k_{12}^{25} \oplus k_{12}^{10}$
3	$k_{12}^3 \oplus k_{12}^{11} \oplus k_{12}^{26}$	18	$k_{12}^{18} \oplus k_{12}^{26} \oplus k_{12}^{11}$
5	$k_{12}^5 \oplus k_{12}^{13} \oplus k_{12}^{28}$	20	$k_{12}^{20} \oplus k_{12}^{28} \oplus k_{12}^{13}$
7	$k_{12}^7 \oplus k_{12}^{15} \oplus k_{12}^{30}$	21	$k_{12}^{21} \oplus k_{12}^{29} \oplus k_{12}^{14}$
8	$k_{12}^8 \oplus k_{12}^0 \oplus k_{12}^{31}$	22	$k_{12}^{22} \oplus k_{12}^{30} \oplus k_{12}^{15}$
9	$k_{12}^9 \oplus k_{12}^1 \oplus k_{12}^{16}$	24	$k_{12}^{24} \oplus k_{12}^{16} \oplus k_{12}^1$
10	$k_{12}^{10} \oplus k_{12}^2 \oplus k_{12}^{17}$	26	$k_{12}^{26} \oplus k_{12}^{18} \oplus k_{12}^3$
12	$k_{12}^{12} \oplus k_{12}^4 \oplus k_{12}^{19}$	27	$k_{12}^{27} \oplus k_{12}^{19} \oplus k_{12}^4$
13	$k_{12}^{13} \oplus k_{12}^5 \oplus k_{12}^{20}$	28	$k_{12}^{28} \oplus k_{12}^{20} \oplus k_{12}^5$
14	$k_{12}^{14} \oplus k_{12}^6 \oplus k_{12}^{21}$	31	$k_{12}^{31} \oplus k_{12}^{23} \oplus k_{12}^8$

The other $22 \times 3 = 66$ expressions corresponding to $l_{11}^{-1}(k_{12}^{32} \parallel k_{12}^{33} \parallel \ldots \parallel k_{12}^{63})$, $l_{11}^{-1}(k_{12}^{64} \parallel k_{12}^{65} \parallel \ldots \parallel k_{12}^{95})$ and $l_{11}^{-1}(k_{12}^{96} \parallel k_{12}^{97} \parallel \ldots \parallel k_{12}^{127})$ are computed in a similar way. Here we have

$$K_{12}^1 = k_{12}^{32} \parallel k_{12}^{33} \parallel \ldots \parallel k_{12}^{63}$$
$$K_{12}^2 = k_{12}^{64} \parallel k_{12}^{65} \parallel \ldots \parallel k_{12}^{95}$$
$$K_{12}^3 = k_{12}^{96} \parallel k_{12}^{97} \parallel \ldots \parallel k_{12}^{127}$$