Side-Channel Attack on ARADI
Bitwise Divide-and-Conquer Key Recovery in Non-Profiled Scenarios

Donggeun Kwon, and Seokhie Hong

School of Cybersecurity, Korea University, South Korea
donggeun.kwon@gmail.com, shhong@korea.ac.kr

Abstract. In this study, we present the first side-channel attack on the ARADI block cipher, exposing its vulnerabilities to physical attacks in non-profiled scenarios. We propose a novel bitwise divide-and-conquer methodology tailored for ARADI, enabling key recovery. Furthermore, based on our attack approach, we present a stepwise method for recovering the full 256-bit master key. Through experiments on power consumption traces from an ARM processor, we demonstrate successful recovery of target key bits, validating the effectiveness of our proposed method. Our findings highlight critical weaknesses in physical security of ARADI and underscore the necessity of implementing effective countermeasures to address side-channel vulnerabilities.

Keywords: Block cipher · Side-channel attack · Non-profiled · Power analysis · ARM processor.

1 Introduction

In modern processor architectures, while the CPU is protected within a secure enclave, the RAM remains vulnerable to physical attacks, increasing the risk of sensitive information being exposed. To mitigate this risk, encryption methods that secure data stored in RAM while ensuring both confidentiality and integrity are essential. Addressing these needs, the U.S. National Security Agency (NSA) proposed a new block cipher for memory encryption, called ARADI in 2024 [4]. Previous block ciphers like AES have struggled to meet these low-latency demands. In contrast, ARADI is designed to achieve efficient performance through hardware optimization, high parallelism, and a reduced number of rounds. However, as with the previously proposed the NSA block ciphers such as SIMON and SPECK, no clear security analysis results or explicit design rationale for ARADI have been disclosed. As a result, ARADI must undergo thorough validation and comprehensive security assessments against a wide range of cryptanalysis before it can be applied in real-world scenarios.

There have been studies evaluating the security of ARADI under various cryptanalytic attacks. Bellini et al. [1] conducted cryptographic analyses by applying multiple cryptanalysis techniques, including differential, linear, integral, and neural distinguishers, using the automated tool CLAASP. In subsequent work, Bellini et al. [2] introduced the concept of *weakly-composed Toffoli gates*, uncovering new structural weaknesses. They proposed novel algebraic distinguishers based on this concept, demonstrating a method to extend existing distinguishers while preserving data complexity. Kim et al. [5] further expanded on the work of Bellini et al. by defining and theoretically proving the byte-wise equal property, showing that it can be extended up to 8 rounds.

Theoretical cryptographic security is a foundational aspect of memory encryption, but it alone is insufficient to protect against practical threats. It is equally important to ensure resistance to various physical attacks, such as side-channel attacks, considering the potential vulnerabilities of memory systems. As these attacks exploit physical characteristics of the implementation, even a robust encryption algorithm can be broken if the implementation is not secure. In this context, it is essential to evaluate the security of the encryption scheme against side-channel analysis.

In this paper, we present the first results of a side-channel attack on the ARADI block cipher conducted in a non-profiled environment. While previous studies primarily focused on assessing its theoretical cryptographic security, our work emphasizes evaluating its physical security in practical implementation scenarios. First, we introduce a key recovery attack on ARADI using correlation power analysis based on a bitwise divide-and-conquer approach, and then, we extend this attack to recover the entire key of ARADI. To validate our proposed technique, we perform key recovery attacks using power consumption data collected from an ARM processor. Our expermental results demonstrate that the block cipher ARADI is indeed vulnerable to side-channel attacks, underscoring the need for more robust countermeasures in its design.

This paper is organized as follows. In Section 2, we briefly introduce the specification of ARADI and provide an overview of non-profiled side-channel analysis. In Section 3, we describe the methodology for conducting side-channel analysis on ARADI in detail, explain each step of the key recovery attack, and verify the proposed approach through experiments. Finally, Section 4 presents our conclusions and discusses directions for future research.

2 Background

In this section, we briefly describe the specifications of the ARADI block cipher targeted in this study and provide an overview of non-profiled side-channel analysis.

2.1 Specification of ARADI

The ARADI block cipher uses a 128-bit block size and a 256-bit key size. Its round function is divided into three main steps: S-box layer, Linear layer, and the round key addition. The encryption function consists of 16 rounds, followed by an additional post-add operation, and is defined as follows.

$$\tau_{rk_{16}} \circ (\Lambda_{15}\pi\tau_{rk_{15}}) \circ \cdots \circ (\Lambda_1\pi\tau_{rk_1}) \circ (\Lambda_0\pi\tau_{rk_0})$$

S-box layer. The S-box layer π of ARADI consists of 4-bit s-boxes, which is designed based on a Toffoli gate. The input state, consisting of four 32-bit words (w, x, y, z), is divided at the bit level so that each bit is independently processed by the S-box. Each bit then undergoes the following four operations in sequence.

$$
\begin{aligned}
x_i &\leftarrow (w_i \wedge y_i) \oplus x_i \\
z_i &\leftarrow (x_i \wedge y_i) \oplus z_i \\
y_i &\leftarrow (w_i \wedge z_i) \oplus y_i \\
w_i &\leftarrow (x_i \wedge z_i) \oplus w_i
\end{aligned}
$$

Linear layer. In Linear layer of ARADI, independent linear transformations are performed on each 32-bit word. The linear layer of the i-th round is denoted as Λ_i, and is represented as follows.

$$\Lambda_i(w, x, y, z) = (L_i(w), L_i(x), L_i(y), L_i(z))$$

L_i is a linear map for 32-bit words, and the transformation changes with each round. The 32-bit input is divided into an upper 16 bits u and a lower 16 bits l, and the following transformation is applied to each word.

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

Here, S_{16} represents a left circular shift operation defined on 16-bit units. The rotation amounts vary by i as shown below.

Table 1. Rotation amounts for i-round.

i mod 4	a_i	b_i	c_i
0	11	8	14
1	10	9	11
2	9	4	14
3	8	9	7

Round key addition and key schedule. The round key addition step is denoted as τ_{rk_i}, where the 128-bit round key rk_i is XORed with the intermediate state of each round. The key schedule takes a 256-bit master key as input and produces a 128-bit round key for each round. To achieve this, the master key is divided into eight 32-bit words (k). Key updating is performed using two linear maps, M_0 and M_1, which transform pairs of 32-bit words. During each round, a different permutation is applied, and a counter value is XORed to prevent slide and rotation attacks. From the generated k_0, k_1, ..., k_7 words for each round, even-numbered rounds use $k_0||k_1||k_2||k_3$ as the round key, while odd-numbered rounds use $k_4||k_5||k_6||k_7$. The two linear maps M_0 and M_1 are defined as follows.

$$
\begin{aligned}
M_0(x, y) &= (S_{32}^{1}(x) \oplus y,\ S_{32}^{3}(y) \oplus S_{32}^{1}(x) \oplus y) \\
M_1(x, y) &= (S_{32}^{9}(x) \oplus y,\ S_{32}^{28}(y) \oplus S_{32}^{9}(x) \oplus y)
\end{aligned}
$$

2.2 Side-Channel Analysis

As the need for security in environments where physical devices are exposed has grown, such as Internet of Things, it has become necessary to ensure not only the mathematical security of cryptographic algorithms but also their resistance to physical attacks. One such physical attack is side-channel analysis, which exploits information leaked when cryptographic algorithms are implemented on actual devices [6], include power consumption, electromagnetic emissions, and execution time.

Depending on attack environments, side-channel attacks can be categorized into profiling attacks and non-profiling attacks. Profiling attacks are possible when the attacker possesses a profiling device identical to the target device. The attacker first characterizes vulnerabilities using this profiling device and then applies the acquired information to analyze the measured waveforms. In contrast, non-profiling attacks do not require a separate profiling device. Instead, the attacker directly collects side-channel data from the target device and employs statistical methods to derive secret information. Examples of non-profiling attacks include Differential Power Analysis (DPA) [7] and Correlation Power Analysis (CPA).

Among the non-profiled side-channel attacks, Correlation Power Analysis, proposed by Brier et al. [3] in 2004, is a statistical method that leverages the correlation between the device's power consumption during cryptographic operations and the intermediate values of those operations. First, the attacker measures the power consumed when the device is performing operations related to the secret information. The attacker then guesses the intermediate values produced by these operations, calculates the leakage, and computes the correlation coefficient between the measured power traces and the guessed intermediate values. The intermediate value guess that yields the highest correlation is assumed to be correct. Given this intermediate value and the known plaintext, the secret key can be recovered. The correlation coefficient between two variables is defined as follows.

$$\rho(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

3 Side-channel Attack on ARADI

In this section, we present a non-profiled side-channel analysis methodology for ARADI and verify its effectiveness through experiments using power consumption. First, we describe how to recover 64 bits of the first-round key. Next, we validate our approach experimentally and then propose a method to recover the remaining 192 bits, ultimately demonstrating a procedure to recover the entire master key of ARADI.

3.1 Bitwise Key Recovery Attack on k_0, k_2

To begin with, we propose a method to recover the 64 bits of the 1-round key, k_0 and k_2. In the S-box layer, the following operation is performed first. We refer this as the target function h_1.

$$h_1 = ((w \oplus k_0) \land (y \oplus k_2)) \oplus (x \oplus k_1)$$

To perform correlation power analysis on ARADI, we focus on a single bit position and consider how the hypothetical intermediate values depend on the bits of the key. Let the i-th bit of k_j be denoted as k_j^i. Depending on the combination of k_0^i, k_1^i, and k_2^i, the hypothetical intermediate output h_1^i is determined as shown in Table 2. Table 2 enumerates all 8 possible 3-bit combinations of (k_0^i, k_1^i, k_2^i) and specifies the corresponding intermediate value that the S-box layer operation would produce under each key-bit scenario.

Table 2. The result of the target function h_1 depending on the cases of the targeted key bits.

k_0^i	k_1^i	k_2^i	h_1^i
0	0	0	$(w \land y) \oplus x$
0	0	1	$(w \land \bar{y}) \oplus x$
0	1	0	$(w \land y) \oplus \bar{x}$
0	1	1	$(w \land \bar{y}) \oplus \bar{x}$
1	0	0	$(\bar{w} \land y) \oplus x$
1	0	1	$(\bar{w} \land \bar{y}) \oplus x$
1	1	0	$(\bar{w} \land y) \oplus \bar{x}$
1	1	1	$(\bar{w} \land \bar{y}) \oplus \bar{x}$

With this setup, an attacker can compute hypothetical intermediate values for all possible bit combinations of (k_0^i, k_1^i, k_2^i). For each combination, the attacker uses the plaintext inputs and the candidate key bits to predict the intermediate value produced during encryption. Subsequently, the attacker calculates the correlation coefficient between the power consumption and the hypothetical value for each case. Based on these calculations, the attacker identifies the correct guess as the one that yields the highest correlation. From this guess, we can recover the i-th bits of k_0^i, k_1^i, k_2^i—a total of three bits.

Since our analysis targets a single bit at a time, the property $a \oplus \bar{b} = \overline{a \oplus b}$ holds. As a result, for k_1, there always exists a pair of cases whose correlations have the same magnitude but opposite signs. For example, the hypothetical value of h_1 for the (0, 0, 0) case is always the bit-flipped value of the (0, 1, 0) case, causing these two cases to produce correlations with opposite signs. This makes it impossible to distinguish which of the two is the correct guess solely based on correlation. However, using this approach, it is still possible to recover k_0^i and

k_2^i. By repeating this anaylsis 32 times for each bit ($i = 1...32$), the 64-bit key, k_0 and k_2, can be obtained using a divide-and-conquer strategy. Moreover, the remaining key bits can also be recovered, and we will discuss this attack method in Section 3.2.

3.2 Recovering Other Key Words

After recovering k_0 and k_2 in the earlier steps, it becomes possible to further dissect the next operations of the S-box layer. Knowing k_0 and k_2 allows us to directly compute the following value $v_1 = (w \oplus k_0) \wedge (y \oplus k_2)$. With v_1, the attacker can now define a new target function h_2 and apply the same approach as described in Section 3.1. The function h_2 is given by:

$$h_2 = ((v_1 \oplus (x \oplus k_1)) \wedge (y \oplus k_2)) \oplus (z \oplus k_3)$$

Since v_1 and k_2 are known, the attacker can perform a 2-bit CPA attack to determine k_1. By recovering k_1 through this analysis, the attacker now possesses a total of three key words—k_0, k_1, and k_2—from the first round. However, in this phase,, k_3 cannot be directly recovered due to the problem of ghost peaks, similar to the phenomenon described in Section 3.3.

Once k_1 is successfully obtained, $v_2 = (v_1 \oplus (x \oplus k_1)) \wedge (y \oplus k_2)$ can be computed and then we proceed to target the h_3 function. With v_2, the attacker can then target h_3 for further analysis:

$$h_3 = ((v_2 \oplus (z \oplus k_3)) \wedge (w \oplus k_0)) \oplus (y \oplus k_2)$$

By conducting a 1-bit CPA attack on h_3, it can be possible to recover k_3. At this point, the attacker obtain four 32-bit words (k_0, k_1, k_2, k_3) that compose the 1-round key rk_0, totaling 128 bits.

Since the first 128 bits of the master key are used directly as the 1-round key in ARADI, acquiring these four words means the attacker effectively recover half of the master key. With the round key rk_0, the intermediate state after the 1-round can be derived, thereby enabling the application of the same attack methodology to the second round. Through this process, the attacker can retrieve the 2-round key, rk_1.

Moreover, since the linear transformations M_0 and M_1 used in the key scheduling are invertible, it is straightforward to compute the remaining 128 bits of the master key, (k_4, k_5, k_6, k_7), from rk_1. In other words, upon completing the analysis for the second round, the attacker can achieve all the information needed to reconstruct the 256-bit key of ARADI.

In summary, after extracting k_0 and k_2, the attacker can derive k_1, and subsequently k_3, thus fully recovering the initial 128-bit round key. This half-key then enables us to attack the second round in the same approach. From there, utilizing the invertibility of M_0 and M_1, you can retrieve the remaining half of the master key. This bitwise divide-and-conquer strategy demonstrates that the entire 256-bit ARADI master key can be recovered step by step through iterative CPA attacks.

3.3 Experimental Results

We conductd experiments to validate the attack proposed in Section 3.1. We collect power consumption measurements at a 20 MS/s sampling rate from an ARM STM32F0 processor running the ARADI encryption. In total, we use 50,000 traces in our experiments. Figure 1 shows the captureed power consumption during the full round encryption of ARADI. In Figure 1, four distinct peaks appear repeatedly 16 times, suggesting that these operations correspond to the round function. We then perform the non-profiled side-channel attack targeting the first round of ARADI.

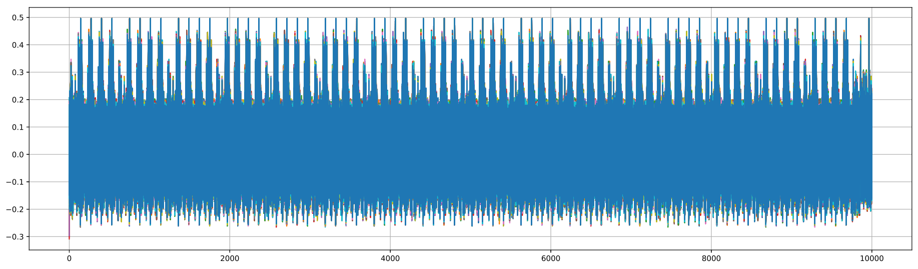
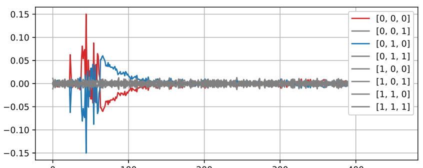


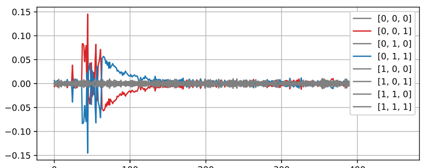
Fig. 1. Power consumption of full-round encryption.

Recovering k_0 and k_2 with target function h_1. We perform CPA attack on all 8 possible 3-bit combinations of (k_0, k_1, k_2). For each combination, we calculate the correlation between the measured power traces and the hypothetical intermediate values, which are derived from the plaintext and candidate keys. The results of our analysis for each case are presented in Figure 2. Figure 2 displays graphs for each (k_0, k_1, k_2) combination, labeled from 2(a) to 2(h). The red plot represents the hypothesis that produces the highest positive correlation peak, while the blue plot corresponds to the hypothesis with the highest negative correlation peak. When a combination appears in red, it means its bit pattern strongly matches the actual key, as shown by a high positive correlation. In contrast, a blue-marked combination shows a similarly strong but oppositely signed correlation. In this situation, the attacker cannot determine whether the relationship between the measured power consumption and the hypothesis is positive or negative. As a result, we cannot distinguish whether the red or the blue case represents the correct guess. As a result, it is impossible to distinguish whether the red or blue case corresponds to the correct guess. However, both cases share k_0 and k_2 with the same values, enabling the attacker to recover those two bits.

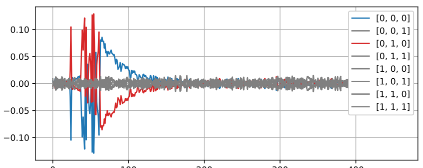
By examining these analysis results, the bits of (k_0, k_2) can be determined relatively easily, as their correct hypothesis consistently yields a highest peak. However, for (k_1), there is always a counterpart with a matching magnitude but negative correlation, making it difficult to distinguish the correct bit. This finding highlights a limitation: k_1 lies between two hypotheses with opposite-sign correlations, revealing the problem of ghost peaks [7]. Despite this, our experimental outcomes confirm the effectiveness of the bitwise approach for k_0 and k_2. We can apply a divide-and-conquer strategy to recover each bit of k_0^i and k_2^i, ultimately enabling extraction of the entire 64-bit key. In other words, by iteratively applying this analysis, an attacker can gradually retrieve the two key words in ARADI.



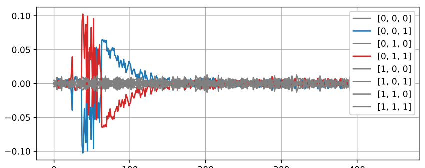
(a) $k_0 = 0, k_1 = 0, k_2 = 0$



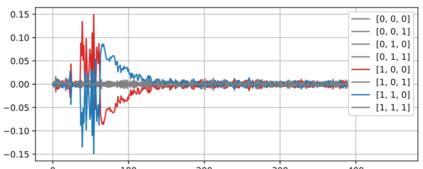
(b) $k_0 = 0, k_1 = 0, k_2 = 1$



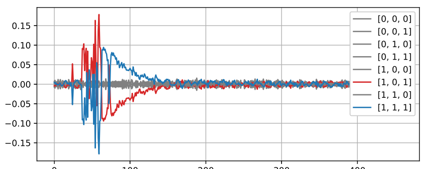
(c) $k_0 = 0, k_1 = 1, k_2 = 0$



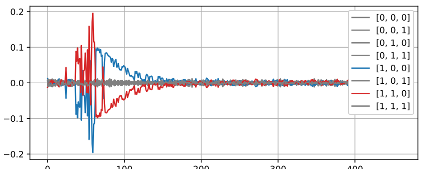
(d) $k_0 = 0, k_1 = 1, k_2 = 1$



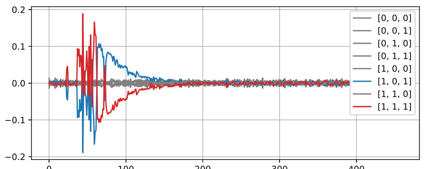
(e) $k_0 = 1, k_1 = 0, k_2 = 0$



(f) $k_0 = 1, k_1 = 0, k_2 = 1$



(g) $k_0 = 1, k_1 = 1, k_2 = 0$



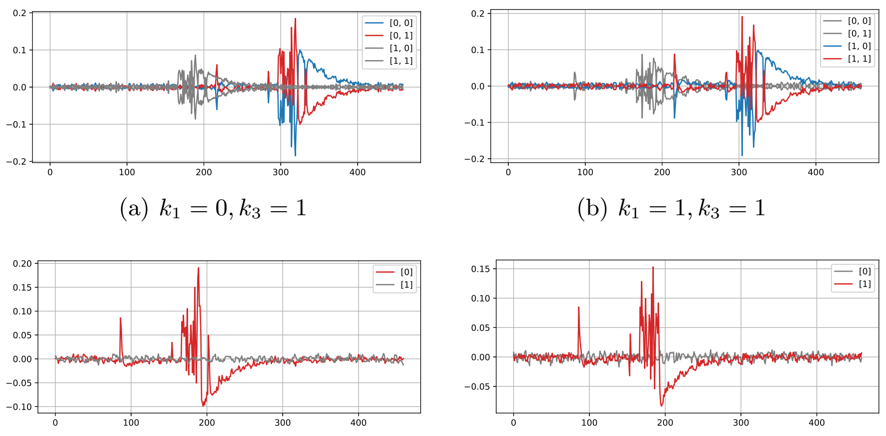
(h) $k_0 = 1, k_1 = 1, k_2 = 1$

Fig. 2. Results of Correlation Power Analysis (red: correct key, blue: ghost key).

Recovering k_1 and k_3 with target functions h_2, h_3. After successfully recovering k_0 and k_2 using the earlier steps, the attacker can proceed to a more refined analysis step that targets the function h_2. In this phase, a 2-bit CPA is performed, using k_0 and k_2 to focus on determining k_1. By evaluating correlation coefficients between the power traces and the hypothetical intermediate values for each candidate key bit combination, the attacker can distinguish which scenario yields the strongest correlation, thereby inferring the correct bit for k_1. Figures 3(a) and 3(b) in Figure 3 provide examples of these results. Each subplot corresponds to a different assumed value of k_1. Specifically, Figure 3(a) shows the case when $k_1 = 0$, and Figure 3(b) shows the case when $k_1 = 1$. By examining the patterns of correlation peaks—both positive and negative peaks—the attacker can confidently deduce the actual bit value of k_1.

As expected from the earlier theoretical considerations in Section 3.2, our experiments also reaffirm the existence of the ghost peaks problem related to k_3. Ghost peaks arise when there are two hypotheses with equal magnitude but opposite sign in their correlation, making it impossible to directly determine k_3 at this stage through the same 2-bit CPA method.

Once k_1 is recovered, the attacker is well-equipped to target the next target function, h_3, for the purpose of recovering k_3. In this phase, a 1-bit CPA is performed. Because k_0, k_1, and k_2 are now known, the analysis can isolate k_3 and overcome the ambiguity that arose during the previous step. Figures 3(c) and 3(d) in Figure 3 illustrate the correlation results for two scenarios: $k_3 = 0$ and $k_3 = 1$, respectively. By comparing these correlation traces and identifying which assumption about k_3 leads to a higher, the attacker can finally recover the correct value of k_3.



(a) $k_1 = 0, k_3 = 1$

(b) $k_1 = 1, k_3 = 1$

(c) $k_3 = 0$

(d) $k_3 = 1$

Fig. 3. Results of Correlation Power Analysis (red: correct key, blue: ghost key).

Overall, these experimental results demonstrate that a step-by-step CPA approach enables the complete recovery of the first-round key. With the first-round key fully determined, the attacker can then apply the same analysis to the second round, thereby extending the strategy to eventually recover the entire master key. Our experimental findings confirm that this divide-and-conquer approach allows iterative CPA attacks to effectively recover the master key of ARADI in a non-profiled environment.

4 Conclusion

In this study, we present the first side-channel attack results on the ARADI block cipher, revealing its vulnerability to power analysis attacks. Through our proposed approach, we successfully demonstrated key recovery attacks that highlight critical weaknesses in ARADI's physical security. Our experimental results validate the effectiveness of our attack methodology, enabling the recovery of the 256-bit master key in the stepwise method. This underscores a significant gap in resistance to physical attacks, emphasizing the urgent need for implementing countermeasures. However, introducing countermeasures may compromise the designers' intended performance, such as its low-latency goal. Future research will focus on the development and application of efficient countermeasures that mitigate these vulnerabilities without significantly reduce memory access speed.

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., Rachidi, M., Rohit, R., Tiwari, S.K.: Mind the composition of toffoli gates: Structural algebraic distinguishers of aradi. Cryptology ePrint Archive (2024)
3. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6. pp. 16–29. Springer (2004)
4. Greene, P., Motley, M., Weeks, B.: Aradi and llama: Low-latency cryptography for memory encryption. Cryptology ePrint Archive (2024)
5. Kim, S., Kim, I., Lee, D., Hong, D., Sung, J., Hong, S.: Byte-wise equal property of aradi. Cryptology ePrint Archive (2024)
6. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16. pp. 104–113. Springer (1996)
7. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. pp. 388–397. Springer (1999)