# Unleashing the Power of Differential Fault Attacks on QARMAv2

Soumya Sahoo[1], Debasmita Chakraborty[2] and Santanu Sarkar[1]

[1] Indian Institute of Technology Madras, Chennai, India
[2] Indian Statistical Institute, Kolkata
soumyasahoo078@gmail.com,debasmitachakraborty1@gmail.com,sarkar.santanu.bir1@gmail.com

**Abstract.** QARMAv2 represents a family of lightweight block ciphers introduced in ToSC 2023. This new iteration, QARMAv2, is an evolution of the original QARMA design, specifically constructed to accommodate more extended tweak values while simultaneously enhancing security measures. This family of ciphers is available in two distinct versions, referred to as QARMAv2-$b$-$s$, where '$b$' signifies the block length, with options for both 64-bit and 128-bit blocks, and '$s$' signifies the key length. In this paper, for the first time, we present differential fault analysis (DFA) of all the QARMAv2 variants- QARMAv2-64, and QARMAv2-128 by introducing an approach to utilize the fault propagation patterns at the nibble level, with the goal of identifying relevant faulty ciphertexts and vulnerable fault positions. This technique highlights a substantial security risk for the practical implementation of QARMAv2. By strategically introducing six random nibble faults into the input of the $(r-1)$-th and $(r-2)$-th backward rounds within the $r$-round QARMAv2-64, our attack achieves a significant reduction in the secret key space, diminishing it from the expansive $2^{128}$ to a significantly more smaller set of size $2^{32}$. Additionally, when targeting QARMAv2-128-128, it demands the introduction of six random nibble faults to effectively reduce the secret key space from $2^{128}$ to a remarkably reduced $2^{24}$. To conclude, we also explore the potential extension of our methods to conduct DFA on various other iterations and adaptations of the QARMAv2 cryptographic scheme. To the best of our knowledge, this marks the first instance of a differential fault attack targeting the QARMAv2 tweakable block cipher family, signifying an important direction in cryptographic analysis.

**Keywords:** Tweakable block cipher, Differential fault attack, Nibble fault, QARMAv2

## 1 Introduction

Fault analysis (FA), initially presented by Boneh et al. [11, 12] targeting RSA-CRT implementations, represents a potent method for attacking cryptographic primitives. This technique enables the attackers to acquire supplementary side-channel data, facilitating the practical attainment of key recovery attacks. Following this milestone, Biham and Shamir [9] introduced differential fault analysis (DFA) on the DES block cipher during CRYPTO 1997. Since then, various fault attacks are introduced such as (statistical) ineffective fault analysis [14, 15], collision fault analysis [10], fault intensity analysis [16], persistent fault attack [26], fault template attack [22], and other fault attacks [17, 21].

The Differential Fault Attack capitalizes on the distinction between accurate and faulty ciphertexts, achieved by introducing faults in the internal state during the final rounds. Using these pairs of correct and faulty ciphertexts, an adversary can employ cryptographic analysis to recover the secret key. A significant aspect of DFA is its capability to analyze a limited number of rounds of a block cipher. DFA has found extensive application in

various attacks targeting the AES [3], TWOFISH [2], PRESENT [7,19], PRINCE [23], MIDORI [13], SKINNY [25] etc.

QARMAv2 is a family of tweakable block ciphers recently proposed by Avanzi et al. [5]. It significantly enhances the original QARMA [4] cipher introduced at FSE 2017. The primary objective of QARMAv2 is to elevate its security capabilities and accommodate longer tweak inputs, all while preserving similar levels of latency and area efficiency. This extended tweak input capacity serves both specific application scenarios and facilitates the development of modes of operation with enhanced security attributes. The advancements in QARMAv2 are realized by introducing novel key and tweak schedules, restructured choices for S-Box and linear layer components, and a more thorough and comprehensive security analysis. In the realm of fully unrolled hardware implementations, QARMAv2 remains competitive by offering a compelling combination of low latency and efficient area utilization. The primary objective of the QARMAv2 design is to create a versatile Tweakable Block Cipher (TBC) that can serve as a general-purpose solution while also excelling in cryptographic memory protection, as demonstrated in [6]. Additionally, it should enable fast computation of short messages, which is particularly useful for ensuring control flow integrity. The design allows for efficient hardware implementations and aims to facilitate the development of optimized software implementations.

Related Works: QARMAv2's design specification includes rigorous security evaluations against multiple cryptanalytic techniques, notably differential, boomerang, integral, impossible differential, and zero-correlation attacks. In the first independent security analysis of QARMAv2, Tim Beyne [8] discovered a nonlinear invariant property within QARMAv2-64's round function when no key is applied. This characteristics persists across multiple rounds only when *weak keys* are used. Therefore, in [18], Hosein et al. has unveiled an integral key recovery attack on QARMAv2. Recently, Ahmadin et al. proposed single-tweak truncated differential attacks on 11-round QARMAv2-64 [1]. Currently, there are no known fault attacks on QARMAv2 in the literature to the best of our knowledge.

## 1.1 Our Contributions

In this paper, we embark on a critical exploration of some interesting properties within QARMAv2 to construct DFA against it, which stems from its diffusion matrix. Our research begins with a thorough examination of the fault propagation characteristics when a single fault is introduced during the $(r-1)$-th backward round within the context of QARMAv2's $r$-round encryption function. Continuing our exploration, we undertake an exhaustive analysis of the difference between correct and faulty ciphertexts, with a particular emphasis on non-zero differential patterns. This idea reveals distinct and interesting patterns that establish a clear connection between faulty positions and the locations of these non-zero differentials. Now, these patterns represent a valuable opportunity for potential exploitation, as they offer insights into the extraction of key bit information. To come up with some more interesting patterns, we also induce a single fault at the beginning of the $(r-2)$-th backward round of QARMAv2 and study the fault propagation characteristics.

Building upon these insights, we introduce a nibble-oriented differential fault analysis targeting both QARMAv2-64 and QARMAv2-128. It's worth noting that these two variants share an almost identical overall structural framework. As our fault model assumes that the attacker lacks knowledge about the specific location of the faulted nibble, we also discuss how the attacker can uniquely determine the fault positions based on some observations. Addition to that, we discover some interesting ideas to construct DFA on QARMAv2 in such a way so that the attack can be performed using optimally minimum number of faults, reducing the secret key space by a huge margin.

Based on our attack methodology, we can reduce the secret key space from $2^{128}$ to $2^{32}$ for QARMAv2-64, and $2^{128}$ to $2^{24}$ for QARMAv2-128-128 by only using six faults. Our

**Table 1:** Summarization of Our Attack

| Variant | Key Size | Number of Distinct Fault Locations | Number of Faults | Complexity |
|---|---|---|---|---|
| QARMAv2-64 | $2^{128}$ | 3 | 6 | $\approx 2^{32}$ |
| QARMAv2-128-128 | $2^{128}$ | 4 | 6 | $\approx 2^{24}$ |
| QARMAv2-128-192 | $2^{192}$ | 5 | 10 | $\approx 2^{40}$ |
| QARMAv2-128-256 | $2^{256}$ | 6 | 12 | $\approx 2^{64}$ |

theoretical analysis has been substantiated through experimental verification, allowing us to successfully recover the keys for QARMAv2-64 and QARMAv2-128-128. The practical attack complexities of these two versions have been confirmed in our experiments. It is noteworthy that, to the best of our knowledge, our work presents the first fault attack on QARMAv2.
The source codes of all our algorithms are available at:

https://github.com/SoumyaSahoo-IITM/DFA-on-QARMAv2.git.

We conclude by outlining how our approach can be adapted to develop DFA for QARMAv2-128-192 and QARMAv2-128-256. Table 1 provides a summary of attack complexities and the requisite number of fault injections across all QARMAv2 variants.

### 1.2 Organization of the Paper

The structure of this paper is as follows: Section 2 details the QARMAv2 cipher and fundamental insights of DFA attacks. Section 3 examines DFA attacks on all QARMAv2 versions, assessing our method's effectiveness. Section 4 delves into complexity analysis and presents simulations that validate our attack strategies. Finally, Section 5 discusses future research avenues and concludes the paper.

## 2 Preliminaries

### 2.1 Brief Description of QARMAv2

In this section, we provide a concise introduction to the QARMAv2 specification. We start by giving a glimpse of the cipher's overall structure and follow it with brief explanations of the sub-operations of the cipher that hold significance within the context of DFA attacks. One may refer to [5] for a detailed description of each sub-operations.

### 2.2 General Description

The construction of QARMAv2 follows the same reflector design as utilized in PRINCE and QARMA. The design consists of a forward function ($F$), a symmetric reflector (referred to as the central construction), and a backward function ($\bar{F}$), which is the functional inverse of the forward function. In Figure 2, we sketch the reflector structure of QARMAv2. The initial and final round consist only of the key addition and substitution layer. In the reflector, tweak is not used. The keys and tweaks are generated from the master key $K$ and

tweak $T$, respectively, using some simple operations. The function $F$ consists of iterated function $R$. Similarly, the function $\bar{F}$ is generated by iterating the round function $\bar{R}$. Here, a bar over a function represents the inverse of the function. A typical round function $R$ is a composition of a state shuffle $\tau$, a diffusion matrix $M$ and a S-Box, i.e., $R = S \circ M \circ \tau$.

The encryption function of $r$-round `QARMAv2` consists of $r$-rounds of the round function $R$ (forward function), the reflector in the middle, $r$-rounds of the round function $\bar{R}$ (backward function), and two half rounds consisting of key addition and substitution layer at the beginning and the end of the encryption function. please refer to Algorithm 1 in [5] for more details.

### 2.2.1 Internal State

The internal state $\mathscr{S}$ of `QARMAv2` is represented by a three-dimensional array comprising $l$ layers where $l$ can be 1 or 2. It is $b$ bits long. The one-layer and two-layer constructions are denoted as `QARMAv2`-64 and `QARMAv2`-128, respectively. A layer can be viewed as a 4 by 4 matrix of 4-bit cells.

$$\mathscr{L} = x_0 \,||\, x_1 \,||\, \cdots \,||\, x_{14} \,||\, x_{15} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix}$$

So, $b = 64l$. For `QARMAv2`-128, $l = 2$, the state is defined as a 8 by 4 matrix.

### 2.2.2 Round Functions

`QARMAv2` reuses the `QARMA` round functions with a single change for the two-layer version. There are four types of round: the *full round* and the *half round*, and their inverses. A full round has the following structure (Figure 1):

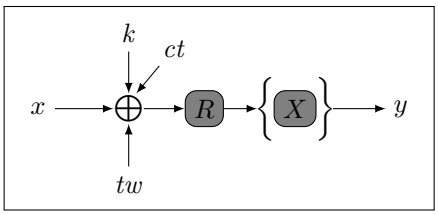



**Figure 1:** A full round of `QARMAv2`

where $R = S \circ M \circ \tau$. The notations $k, ct, tw$ represent a round key, constant, and tweak, respectively. The symbol $\tau$ signifies the `MIDORI StateShuffle` operation. The matrix $M$ is a $4 \times 4$ matrix that conducts left multiplication on each layer of a block, working column-wise. Additionally, the symbol $S$ denotes the simultaneous application of identical S-Boxes to all cells of the state. Please note that comprehensive definitions for these components will be provided later in this section.

In Figure 1, the letter $X$ represents the `eXchangeRows` operation, specifically applicable when $l = 2$. This operation involves swapping the first two rows between two layers. It occurs every other full round, with two `eXchangeRows` operations always flanking the reflector, or, in simpler terms, `eXchangeRows` is consistently included in Rounds $r$ and $r + 1$. It is worth noting that both $X$ and $S$ operations exhibit compatibility and can be executed in any sequence.

### 2.2.3 Reflector Function

The reflector (or central construction) is similar to that of QARMA (Figure 2):

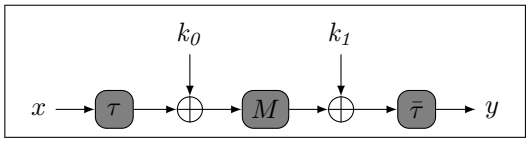



**Figure 2:** Reflector function of QARMAv2

where $k_0$, $k_1$ are two round keys. One can achieve the implementation of this function by utilizing the original QARMA central construction, wherein the round key is derived as $M \cdot k_0 + k_1$. It's worth noting that the reflector component, devoid of S-Box layers, is not considered as a round in this context.

### 2.2.4 Round Keys

For $l = 1$, the admissible key size is 128 bits, and the master key is denoted by $K = K_0||K_1$. The key schedule for the forward and backward functions can be implemented by two registers $k_0$ and $k_1$ that are alternatingly added to that state. Encryption is initialized with values $K_0$ and $K_1$, two halves of the key $K$. The two round keys added in the reflector are $W_0 = o^2(K_0)$ and $W_1 = o^{-2}(K_1)$ where $o$ is an orthomorphism defined as $o : \mathbb{F}_2^b \to \mathbb{F}_2^b$ such that:

$$o(w) := (w \ggg 1) + (w \gg (b-1)).$$

During the computation of the backward function, $K_0$ and $K_1$ are subject to the transformation

$$\iota_e : (K_0, K_1) \mapsto (L_0, L_1) := (o(K_0) + \alpha, o^{-1}(K_1 + \beta)).$$

If $k_0$ and $k_1$ are instead initially set to the values $L_0 = o(K_0) + \alpha$ and $L_1 = o^{-1}(K_1) + \beta$, the inverse transformation is given by

$$\iota_d : (L_1, L_0) \mapsto (o(L_1) + o(\beta), o^{-1}(L_0) + o^{-1}(\alpha)) = (K_1, K_0).$$

Thus, for decryption we start with $L_1 \mapsto k_0$ and $L_0 \mapsto k_1$ and apply $\iota_d$ instead of $\iota_e$, which differ only in the constants.

*Remark* 1. For the detailed description of round constants and the round tweaks, please refer to [5].

## 2.3 Specification of Sub-operations

The specification of the forward function $F$ describes iterative rounds consisting of five sub-operations (in order) - `AddRoundKey (ARK)`, `AddRoundTweaks and Constants (ART)`, `StateShuffle (`$\tau$`)`, `MixColumns (`$M$`)`, and `SubstitutionLayer (`$S$`)`. The backward function $\bar{F}$ consists of five sub-operations (in order) - `Inverse SubstitutionLayer (`$\bar{S}$`)`, `Inverse MixColumns (`$\bar{M}$`)`, `Inverse StateShuffle (`$\bar{\tau}$`)`, `AddRoundTweaks and Constants (ART)`, and `AddRoundKey (ARK)`.

### 2.3.1 State Shuffle

MIDORI's shuffle $\tau$ is used as a state shuffle in QARMAv2 construction.

$$\tau = [\ 0, 11, 6, 13, 10, 1, 12, 7, 5, 14, 3, 8, 15, 4, 9, 2]$$

$\tau$ operates on each layer in the following manner:

$$\mathscr{L} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix} \xrightarrow{\tau} \begin{pmatrix} a_0 & a_{11} & a_6 & a_{13} \\ a_{10} & a_1 & a_{12} & a_7 \\ a_5 & a_{14} & a_3 & a_8 \\ a_{15} & a_4 & a_9 & a_2 \end{pmatrix} = \tau(\mathscr{L})$$

The state shuffle inverse $\tau^{-1}$ is defined as

$$\tau^{-1} = [\ 0, 5, 15, 10, 13, 8, 2, 7, 11, 14, 4, 1, 6, 3, 9, 12]$$

and its operates on each layer as follows:

$$\mathscr{L} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix} \xrightarrow{\tau^{-1}} \begin{pmatrix} a_0 & a_5 & a_{15} & a_{10} \\ a_{13} & a_8 & a_2 & a_7 \\ a_{11} & a_{14} & a_4 & a_1 \\ a_6 & a_3 & a_9 & a_{12} \end{pmatrix} = \tau^{-1}(\mathscr{L})$$

At any $i$th forward round, $\tilde{W}^i$ and $\tilde{Z}^i$ present the input and the output state of the State Shuffle operation, and at the $i$th backward round, $Z^i$ and $W^i$ present the input and the output state of the Inverse State Shuffle operation. Following the notational conventions of this paper, the State Shuffle operation can be represented as

$$\begin{cases} \tilde{Z}^i = \tau(\tilde{W}^i), \text{ with } 1 \leq i \leq r \\ W^i = \bar{\tau}(Z^i), \text{ with } 1 \leq i \leq r \end{cases}$$

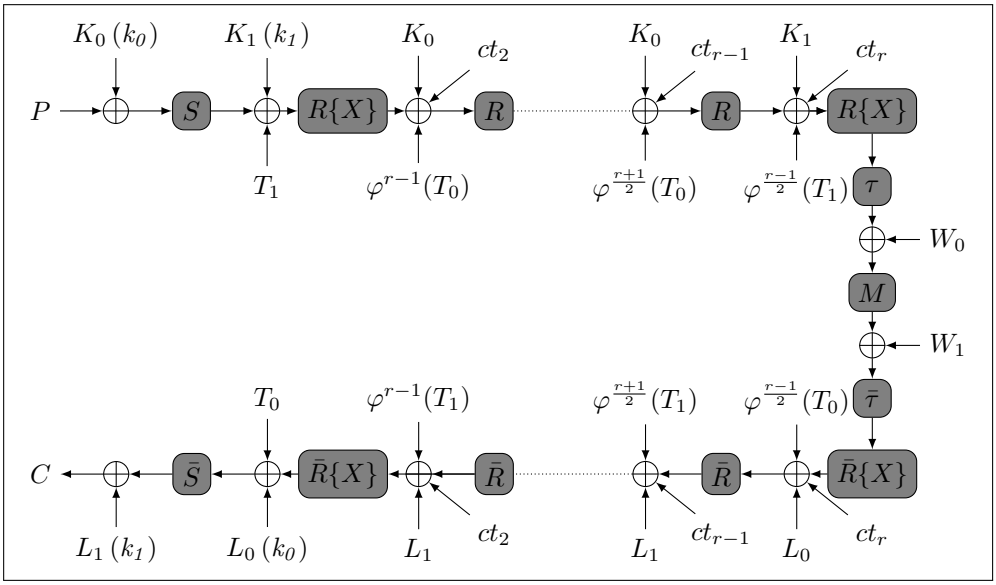



**Figure 3:** `QARMAv2` encryption for odd r. If r is even, $W_0$ and $W_1$ are swapped and the forward function starts with $R$ instaed of $R\{X\}$.

### 2.3.2 Diffusion Matrix

Define a linear transformation $\rho$ on $\mathbb{F}_2^4$ by $\rho(x_3, x_2, x_1, x_0) = (x_2, x_1, x_0, x_3)$ and $\rho^4 = I$ where $I$ is the identity map. Now, the diffusion matrix M is defined using the left cyclic rotation function $\rho$ by

$$M = \text{circ}(0, \rho, \rho^2, \rho^3) = \begin{pmatrix} 0 & \rho & \rho^2 & \rho^3 \\ \rho^3 & 0 & \rho & \rho^2 \\ \rho^2 & \rho^3 & 0 & \rho \\ \rho & \rho^2 & \rho^3 & 0 \end{pmatrix}$$

For the involution property of $M$, we can write $\bar{M} = M$. At any $i$th forward round, $\tilde{Z}^i$ and $\tilde{Y}^i$ present the input and the output state of the MixColumns operation, and at the $i$th backward round, $Y^i$ and $Z^i$ present the input and the output state of the Inverse MixColumns operation. Following the notational conventions of this paper, the MixColumns operation can be represented as

$$\begin{cases} \tilde{Y}^i = M(\tilde{Z}^i), \text{ with } 1 \leq i \leq r \\ Z^i = \bar{M}(Y^i), \text{ with } 1 \leq i \leq r \end{cases}$$

### 2.3.3 Substitution Layer

The Substitution layer sub-operation applies a non-linear bijective transformation on each cell of the internal layer $\mathscr{L}$. The following S-Box is used for `QARMAv2` construction.

$$S = [\ 4\ 7\ 9\ B\ C\ 6\ E\ F\ 0\ 5\ 1\ D\ 8\ 3\ 2\ A\ ]$$

Optionally, one more S-Box is allowed for the PAC and memory authentication applications.

$$\sigma = [\ 0\ E\ 2\ A\ 9\ F\ 8\ B\ 6\ 4\ 3\ 7\ D\ C\ 1\ 5\ ]$$

At any $i$th forward round, $\tilde{Y}^i$ and $\tilde{X}^{i+1}$ present the input and the output state of the S-Box operation, and at the $i$th backward round, $X^i$ and $Y^i$ present the input and the output state of the Inverse S-box operation. Following the notational conventions of this paper, the S-box operation can be represented as

$$\begin{cases} \tilde{X}_j^{i+1} = S(\tilde{Y}_j^i), \text{ with } 1 \leq i \leq r,\ 0 \leq j \leq 15 \\ Y_j^i = \bar{S}(X_j^i), \text{ with } 1 \leq i \leq r,\ 0 \leq j \leq 15 \end{cases}$$

Hence, according to the notational description written above, we can simply describe all the sub-operations involved in the $i$th forward and backward rounds as follows:

$$\begin{cases} \tilde{X}^i \xrightarrow{ART} \tilde{U}^i \xrightarrow{ARK} \tilde{W}^i \xrightarrow{\tau} \tilde{Z}^i \xrightarrow{M} \tilde{Y}^i \xrightarrow{S} \tilde{X}^{i+1} \\ X^i \xrightarrow{\bar{S}} Y^i \xrightarrow{\bar{M}} Z^i \xrightarrow{\bar{\tau}} W^i \xrightarrow{ARK} U^i \xrightarrow{ART} X^{i+1} \end{cases}$$

Similarly, for the encryption of `QARMAv2`-128, we simply use the following notations in order to describe all the sub-operations involved in the $i$th backward rounds:

$$X^i \xrightarrow{X} X'^i \xrightarrow{\bar{S}} Y^i \xrightarrow{\bar{M}} Z^i \xrightarrow{\bar{\tau}} W^i \xrightarrow{ARK} U^i \xrightarrow{ART} X^{i+1}$$

where '$X$' is the `eXchangeRows` operation in `QARMAv2`-128.

## 2.4 Differential Fault Analysis

Differential Fault Analysis (DFA) is a sophisticated side-channel attack that amalgamates traditional differential analysis with advanced fault injection techniques. This form of attack thrives on the discrepancies in information that are induced by deliberate fault injections into a cryptographic device during its operation.

The complexity of executing a DFA is largely determined by the nature of the fault injected. The fault models can be categorized based on two factors: the scale of fault-injection and the ability to position the injected fault. Each of these categories is further divided as follows:

**Scale of Fault-Injection:**

1. *Bit Flip:* Here, the attacker can manipulate a single bit within a specific word.
2. *Byte Error:* In this case, the attacker can modify a single byte within a specific word, substituting it with a random value.
3. *Word Error:* This involves the attacker changing an entire word within the cryptographic device to a random value.

**Positioning Capability:**

1. *Chosen Position:* This implies that the attacker can accurately specify the location for fault injection within the system.
2. *Random Position:* In this situation, the attacker lacks control over the precise location of fault injection; instead, faults manifest randomly within the system.

In cases where the fault-injection scale is smaller, the attacker's assumptions become stronger. Additionally, when a specific fault injection position can be determined, the attacker's assumptions gain strength. Fault injection attacks can be carried out using a range of methods, such as low-hammer attacks, laser-induced faults, electromagnetic fault injection, and other techniques.

# 3 Differential Fault Analysis of QARMAv2

The fundamental principle underlying Differential Fault Analysis (DFA) attacks involves deliberately introducing a targeted fault into a computational process and subsequently examining the XOR (exclusive OR) discrepancy between the results of the correct computation and the flawed computation. Analysts can identify patterns that reveal sensitive information about the cryptographic algorithm or key being used by comparing the expected correct ciphertext with the ciphertext generated when a fault is introduced.

**Table 2:** Differential Distribution Table for S-box of QARMAv2

| Differential | | Output | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | *A* | *B* | *C* | *D* | *E* | *F* |
| Input | 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 2 | 2 | 2 | 0 | 2 | 0 | 0 | 2 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| | 2 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 2 | 0 | 2 | 2 | 0 | 0 |
| | 3 | 0 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 2 | 2 |
| | 4 | 0 | 2 | 0 | 2 | 2 | 0 | 2 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 2 |
| | 6 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 2 | 2 | 2 |
| | 7 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 4 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 2 |
| | 8 | 0 | 0 | 2 | 0 | 4 | 4 | 2 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| | 9 | 0 | 2 | 0 | 0 | 4 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 2 | 2 | 2 |
| | *A* | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 4 | 0 | 4 | 0 |
| | *B* | 0 | 0 | 0 | 0 | 2 | 0 | 4 | 2 | 0 | 2 | 0 | 2 | 2 | 2 | 0 | 0 |
| | *C* | 0 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 0 | 0 | 2 |
| | *D* | 0 | 0 | 0 | 4 | 0 | 0 | 2 | 2 | 0 | 4 | 0 | 0 | 0 | 0 | 2 | 2 |
| | *E* | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 2 | 0 | 4 | 2 | 0 |
| | *F* | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 2 |

DFA attacks on block ciphers often utilize fault difference equations at non-linear components like S-Boxes to deduce secret keys. Formally, we seek to solve for variable $x$ in equations: $SC(x+\delta_i)+SC(x) = \delta_o$ for forward operations, and $SC^{-1}(x+\delta_o)+SC^{-1}(x) = \delta_i$ for reverse operations. In these equations, $\delta_i$ and $\delta_o$ are the input and output differentials at the non-linear stages, respectively, with $SC$ representing a typical non-linear S-Box. The "+" operator denotes binary XOR throughout this paper, and these equations apply symmetrically to both forward and inverse operations. Within DFA, $x$ typically indicates a segment of the undisclosed key, with solutions primarily sought in the inverse S-Box or fault difference equations.

In our scenario, the fault difference equations consist of S-Box equations rather than inverse S-Box ones. Multiple equations of this kind can be formulated for each S-Box in question. The main objective is to effectively solve this set of equations to retrieve the secret keys. While the output differentials $\delta_o$ are known, the input differentials $\delta_i$ may not be fully revealed to the adversary. Nevertheless, the adversary must have at least some partial knowledge of $\delta_i$ to differentiate between valid and invalid key guesses.

Determining the number of key solutions in fault difference equations is crucial. S-Boxes often exhibit non-bijective behavior with input differentials, leading to Differential Distribution Tables (DDTs). Consequently, a single input-output differential may have multiple $x$ solutions, or some differentials may lack solutions entirely, termed impossible differentials. DFA computes the average solution count for a differential equation across the entire DDT to handle scenarios with multiple solutions. When both $\delta_i$ and $\delta_o$ are precisely known, impossible differentials are avoided. In such instances, the average is computed over the non-zero section of the DDT [24]. Table 2 presents the DDT for the S-Box of `QARMAv2`.

**Attack Idea.** Our key recovery attack focuses on some important observations found while investigating the structure of `QARMAv2`, which actually stems from its diffusion matrix. In particular, when we deliberately introduce a single fault during the $(r-1)$-th backward round of `QARMAv2`, we discover interesting connections between faults in different stages, exploiting the structure of the diffusion matrix. We select these connections strategically to create a set of equations that involve the unknown key components. To gain a better understanding of these unknown key components, and specifically to reduce the complexity of our attack, we also introduce a fault at the beginning of the $(r-2)$-th backward round. This helps us to further analyze the impact of faults on the security of the encryption process.

Now, we elaborate on crucial observations that significantly contribute to the construction of DFA on `QARMAv2` with the minimal number of faults, outlined as follows.

**Observation 1** *If we inject two faults at two distinct locations at the beginning of the $(r-1)$-th backward round of `QARMAv2-64`, we can retrieve* 14 *cells of the key $L_1$.*

We conducted observations indicating that performing four fault injections at two distinct locations at the beginning of the $(r-1)$-th backward round allows for the recovery of the key $L_1$ of `QARMAv2`-64 with a complexity of $2^8$. It is crucial to note the significance of the chosen round; injecting faults at the commencement of either the $r$-th backward round or the $(r-k)$-th backward round, where $k \geq 2$, does not enable the retrieval of the key $L_1$ with only two faults within the complexity of $2^8$. In these instances, the number of obtained relations after the fault injections is considerably less, necessitating a greater number of faults compared to those required for the initial fault injection in the backward round $(r-1)$ to recover the key $L_1$. Consequently, we opt for the $(r-1)$-th backward round for the initial fault injection to recover $L_1$ in `QARMAv2`-64. It is noteworthy that this observation holds true for all other variants of `QARMAv2`.

**Observation 2** *After recovering the key $L_1$, if we inject one fault at the beginning of the $(r-2)$-th backward round of QARMAv2-64, we can retrieve information pertaining to all cells of the key $L_0$.*

In our pursuit of recovering the QARMAv2-64 key $L_0$, we initially introduced faults at the start of the $(r-2)$-th backward round. However, upon observation, we found a more effective strategy: leveraging the output from faults injected at the beginning of the $(r-1)$-th backward round, utilizing information from key $L_1$. This strategic adjustment enables us to minimize the number of fault injections required at the outset of the $(r-2)$-th backward round while effectively recovering $L_0$. Based on these important observations, we actually propose the first differential fault attack on all variants QARMAv2 using the minimum number of faults and also reducing the secret key space by a huge margin. For all the variants, we have assumed that the round constants and tweaks are not present because they are already known to the attacker and do not play any role in the differential model. In the ensuing subsection, we outline the attack model. Following that, we detail key recovery attacks on two versions of QARMAv2: QARMAv2-64 and QARMAv2-128. Additionally, we explore the possibility of extending our techniques to perform DFA on other versions of QARMAv2.

## 3.1 Fault Model

The paper's attacks rely on the following assumptions:

- The adversary can inject arbitrary nibble faults at some particular round of QARMAv2. This assumption is deemed practical and essential for DFA attacks. It is important to note that, QARMAv2 is indeed designed to be implemented in the unrolled fashion, and finding the exact position to inject the fault would be difficult. While unrolled implementations present fault injection challenges, recent advanced fault injection techniques overcome these limitations. Interestingly, the occurrence timing of specific computational events is frequently derivable from a suitable side channel leakage. More precisely, the attacker can leverage the power consumption profile to estimate the total time for one encryption and precisely inject the fault at around $((r-1)/r)$-th time for injecting fault in $(r-1)$-th round, where $r$ is the total number of rounds in QARMAv2. These advancements justify our assumption that the attacker can determine computational phases equivalent to specific rounds in the case of a fully unrolled implementation, and can inject random nibble faults at a particular round of QARMAv2[1].
- Furthermore, it is presumed that the attacker is unaware of the precise byte or nibble location affected by the fault. This assumption is also considered reasonable and accommodating in the realm of fault-based attacks.

## 3.2 Key Recovery Attack of QARMAv2-64

Let a nibble fault be injected at the first cell of the QARMAv2-64 state at the beginning of the $(r-1)$-th backward round. Here, we consider the round after the reflector, and the beginning of the $(r-1)$-th backward round refers to the position before three S-Boxes from the last round of the encryption algorithm. The fault goes through twice a inverse round function $\bar{R}$ followed by a inverse S-Box $\bar{S}$ where $\bar{R} = \bar{\tau} \circ \bar{M} \circ \bar{S}$. The propagation of the fault $f$ is depicted in Fig. 5. In Fig. 5, we have introduced some notations. After applying $\bar{S}$ on the fault $f$, it converted to $f'$. Then, the MixColumn operation distributes the fault in three cells. Here $f'_i = \rho^i(f')\ \forall\ 1 \leq i \leq 3$. StateShuffle is applied to the

[1]This model mirrors the validated fault attack model for PRINCE [20, 23], another low-latency cipher employing unrolled architectures.

faults. Application of ARK and ART does affect the faults. This completes one inverse round function $\bar{R}$. Similarly, one more inverse round function $\bar{R}$ is applied to the state where we have introduced more variables (like $l, j, g, h$ with their subscripts). Finally, we apply the inverse S-Box and ARK to get $a_1, a_2, a_3, b_1, b_2, b_3$ and $d_1, d_2, d_3$, which help us to create a set of equations for our key recovery attacks. The remaining notations are explained whenever needed.

### 3.2.1 Identifying the Fault Injection Location

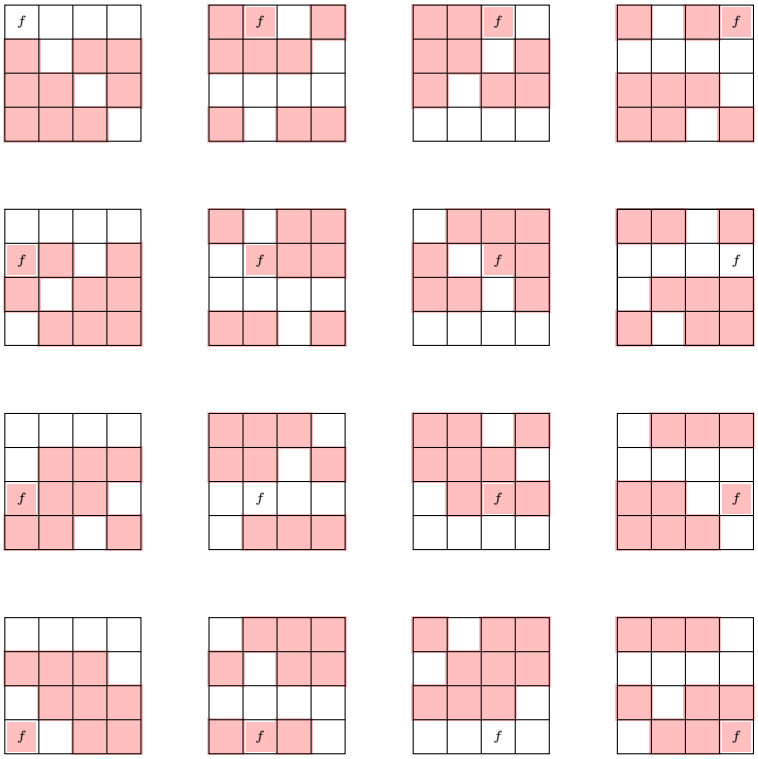


**Figure 4:** Fault propagation pattern in QARMAv2-64 for each fault location at the beginning of the $(r-1)$-th backward round. $f$ denotes the fault injection location and the coloured cells denote the fault diffusion pattern up to and including the inverse S-Box of the last half round function.

Fig. 4 describes all the fault propagation patterns for all 16 possible fault injections at the beginning of the $(r-1)$-th backward round. The patterns are computed up to the last round. An interesting observation is that all the patterns are distinct, corresponding to a distinct fault position. Due to this distinctness, an attacker who only have the knowledge that one fault is injected at the beginning of the $(r-1)$-th backward round, the attacker can uniquely determine the fault location. Moreover, the uniformity of the patterns gives more relaxation to choose the fault position in most cases.

### 3.2.2 $L_1$ Recovery

From Fig. 4, it can be observed that injecting the fault $f$ at any location from all possible 16 locations gives the same number of affected cells. Another important observation is that choosing random faults at any two distinct locations always affects exactly but different 14 cells. Again, the distinctness of all the patterns gives the information about in which

cell the initial fault was injected, assuming that the random fault was initially injected at the beginning of the $(r-1)$-th backward round. However, the attacker does not know the fault injection cell. For the $L_1$ key recovery of QARMAv2-64, we need random fault injection at two randomly chosen cell locations. For the discussion, we have assumed that the faults are injected at $0^{th}$ and $1^{st}$ position of the state at the beginning of the $(r-1)$-th backward round, i.e., at $X_0^{r-1}$ and $X_1^{r-1}$. Fig. 5 shows the fault propagation with fault induced at $0^{th}$ cell. The last grid denotes the cipher text differential $\Delta C$

$$\Delta C = \begin{pmatrix} 0 & 0 & 0 & 0 \\ b_2 & 0 & d_2 & a_2 \\ a_1 & d_3 & 0 & b_1 \\ d_1 & a_3 & b_3 & 0 \end{pmatrix}.$$

Denote the original ciphertext text originating from random plaintext by $C$ and the faulty ciphertext originating from the same plaintext using the same key with fault $f$ at $X_0^{r-1}$ by $C^*$ where $C^* = C + \Delta C$ and

$$C = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{pmatrix}, \qquad C^* = \begin{pmatrix} c_0^* & c_1^* & c_2^* & c_3^* \\ c_4^* & c_5^* & c_6^* & c_7^* \\ c_8^* & c_9^* & c_{10}^* & c_{11}^* \\ c_{12}^* & c_{13}^* & c_{14}^* & c_{15}^* \end{pmatrix}.$$

During the computation of the backward function, two round keys, $L_0$ and $L_1$, are used attentively. This section discusses the key recovery of $L_1$, which is XORed with $Y^{r-1}$, just before the last grid in Fig.5.

$$L_1 = \begin{pmatrix} k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \end{pmatrix}.$$

In section 2.2.4, we have mentioned how the round keys $L_0$ and $L_1$ are generated from the master key $K = K_0 || K_1$. Also, this transformation is invertible, i.e., recovering the keys $L_0$ and $L_1$ is equivalent to recovering the master key $K = K_0 || K_1$. For this reason, we only concentrate on the recovery of $L_0$ and $L_1$.

**Fault Injection at $X_0^{r-1}$.** Referring to the Fig. 5, lets consider the differential state $\Delta U^r$. The same coloured differences are interrelated. We will use these relations to construct a set of equations in unknown key variables. Here, the interrelations between the cells are given.

$$\Delta U^r = \begin{pmatrix} 0 & 0 & 0 & 0 \\ g_2 & 0 & h_2 & j_2 \\ j_1 & h_3 & 0 & g_1 \\ h_1 & j_3 & g_3 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \rho^2(l_2) & 0 & \rho^2(l_3) & \rho^2(l_1) \\ \rho(l_1) & \rho^3(l_3) & 0 & \rho(l_2) \\ \rho(l_3) & \rho^3(l_1) & \rho^3(l_2) & 0 \end{pmatrix}.$$

The variables $l_1, l_2$ & $l_3$ are as in Fig. 5. The variables $j_1, j_2$ & $j_3$ are related by the relation $\rho(j_2) = j_3 = \rho^2(j_1)$, where

$$\begin{cases} j_1 = S(c_8 + k_8) + S(c_8^* + k_8) \\ j_2 = S(c_7 + k_7) + S(c_7^* + k_7) \\ j_3 = S(c_{13} + k_{13}) + S(c_{13}^* + k_{13}) \end{cases}$$

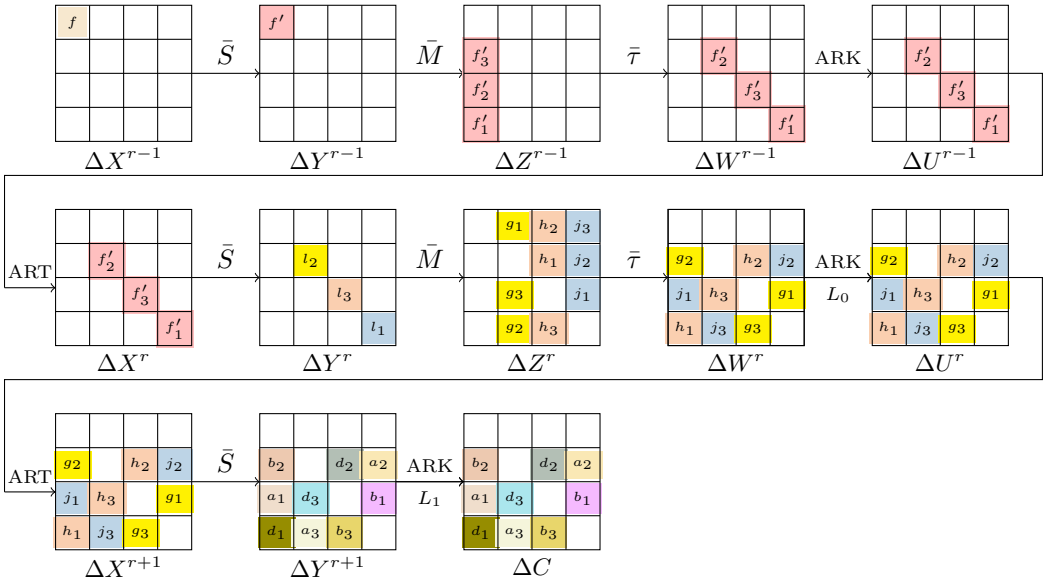



**Figure 5:** Fault propagation in QARMAv2-64 with fault induced at $0^{th}$ cell at the beginning of the $(r-1)$-th backward round. Each variable represents a non-zero fault value and an empty cell corresponds to a zero differential.

These relations give a set of two linearly independent equations such as Eqn. (1) which involve the keys $k_7$, $k_8$ & $k_{13}$.

$$\begin{cases} \rho[S(c_7+k_7)+S(c_7^*+k_7)] &= S(c_{13}+k_{13})+S(c_{13}^*+k_{13}) \\ \rho^2[S(c_8+k_8)+S(c_8^*+k_8)] &= S(c_{13}+k_{13})+S(c_{13}^*+k_{13}) \end{cases} \tag{1}$$

Now, we have two independent equations with three unknown key variables $k_7$, $k_8$ & $k_{13}$. Solving these two equations, we will get approximately $2^4$ solutions. Due to the non-bijective nature of S-Box for a given differential input, the number of solutions will not be exactly $2^4$ but approximately $2^4$. This approximately reduces the key space $2^{12}$ to $2^4$. Changing the fault value once, the keys $k_7$, $k_8$ & $k_{13}$ are determined uniquely with probability almost one. Similarly, we can deduce two more sets of equations. From the relation $\rho(g_2) = g_3 = \rho^2(g_1)$, and from $\rho(h_2) = h_3 = \rho^2(h_1)$, the following set of equations are generated.

$$\begin{cases} \rho[S(c_4+k_4)+S(c_4^*+k_4)] &= S(c_{14}+k_{14})+S(c_{14}^*+k_{14}) \\ \rho^2[S(c_{11}+k_{11})+S(c_{11}^*+k_{11})] &= S(c_{14}+k_{14})+S(c_{14}^*+k_{14}) \\ \rho[S(c_6+k_6)+S(c_6^*+k_6)] &= S(c_9+k_9)+S(c_9^*+k_9) \\ \rho^2[S(c_{12}+k_{12})+S(c_{12}^*+k_{12})] &= S(c_9+k_9)+S(c_9^*+k_9) \end{cases} \tag{2}$$

As in the case of Eqn. (1), from Eqn. (2), the keys $k_4$, $k_{11}$ & $k_{14}$; and $k_6$, $k_9$ & $k_{12}$ are determined uniquely with probability almost one. Therefore, due to fault injection at $X_0^{r-1}$ we have the 9 unique keys $k_4$, $k_6$, $k_7$, $k_8$, $k_9$, $k_{11}$, $k_{12}$, $k_{13}$ & $k_{14}$ out of 16.

**Fault Injection at $X_1^{r-1}$.** Similarly, one more fault injection at $X_1^{r-1}$ generates 6 more independent equations written in Eqn. (3), and by changing the fault value once we can retrieve uniquely nine key cells of $L_1$ which are precisely $k_0$, $k_1$, $k_3$, $k_4$, $k_5$, $k_6$, $k_{12}$, $k_{14}$ & $k_{15}$. Note that in Eqn. (3), some of the key variables are already retrieved from the fault injection at $X_0^{r-1}$. This makes the key recovery easier.

$$\begin{cases} \rho[S(c_{15}+k_{15})+S(c_{15}^*+k_{15})] &= S(c_5+k_5)+S(c_5^*+k_5) \\ \rho^2[S(c_0+k_0)+S(c_0^*+k_0)] &= S(c_5+k_5)+S(c_5^*+k_5) \\ \rho[S(c_{12}+k_{12})+S(c_{12}^*+k_{12})] &= S(c_6+k_6)+S(c_6^*+k_6) \\ \rho^2[S(c_3+k_3)+S(c_3^*+k_3)] &= S(c_6+k_6)+S(c_6^*+k_6) \\ \rho[S(c_{14}+k_{14})+S(c_{14}^*+k_{14})] &= S(c_1+k_1)+S(c_1^*+k_1) \\ \rho^2[S(c_4+k_4)+S(c_4^*+k_4)] &= S(c_1+k_1)+S(c_1^*+k_1) \end{cases} \tag{3}$$

Combining the solutions for the faults at $X_0^{r-1}$ and $X_1^{r-1}$, we have 14 unique keys $k_0$, $k_1$, $k_3$, $k_4$, $k_5$, $k_6$, $k_7$, $k_8$, $k_9$, $k_{11}$, $k_{12}$, $k_{13}$, $k_{14}$ & $k_{15}$ out of 16. We are left with two keys $k_2$ & $k_{10}$. A brute force search can be done for these two keys.

#### 3.2.3 $L_0$ Recovery

For the simplicity of notations here again we represent the key $L_0$ as

$$L_0 = \begin{pmatrix} k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \end{pmatrix}.$$

For $L_0$ key recovery, we will take advantage of the faults at $\Delta X_0^{r-1}$ and $\Delta X_1^{r-1}$, which was earlier induced for the key recovery of the key $L_1$.

**Relations for random fault at $\Delta X_0^{r-1}$.** As we already know the key $L_1$, referring to the Fig. 5, for fault at $\Delta X_0^{r-1}$ we know the original $U^r$ and faulty $U^{r*}$ with fault $\Delta U^r$. These are the states just after the addition of the round key $L_0$, i.e., the fourth grid from the last in Fig. 5. We define

$$U^r = \begin{pmatrix} u_0 & u_1 & u_2 & u_3 \\ u_4 & u_5 & u_6 & u_7 \\ u_8 & u_9 & u_{10} & u_{11} \\ u_{12} & u_{13} & u_{14} & u_{15} \end{pmatrix}, \qquad U^{r*} = \begin{pmatrix} u_0^* & u_1^* & u_2^* & u_3^* \\ u_4^* & u_5^* & u_6^* & u_7^* \\ u_8^* & u_9^* & u_{10}^* & u_{11}^* \\ u_{12}^* & u_{13}^* & u_{14}^* & u_{15}^* \end{pmatrix}.$$

To recover the key $L_0$, we will concentrate on the fifth grid $\Delta U^{r-1}$ in Fig. 5. The three non-zero differentials in this grid are interrelated. Again, we will use these relations to construct a set of equations in unknown key variables. Here, the interrelations between the cells are given.

$$\Delta U^{r-1} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & f_2' & 0 & 0 \\ 0 & 0 & f_1' & 0 \\ 0 & 0 & 0 & f_3' \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \rho^2(f') & 0 & 0 \\ 0 & 0 & \rho^3(f') & 0 \\ 0 & 0 & 0 & \rho(f') \end{pmatrix}.$$

Where

$$\begin{cases} f_1' = S(\rho(u_{13}+k_{13})+\rho^2(u_7+k_7)+\rho^3(u_8+k_8))+S(\rho(u_{13}^*+k_{13})+\rho^2(u_7^*+k_7)+\rho^3(u_8^*+k_8)) \\ f_2' = S(\rho(u_{14}+k_{14})+\rho^2(u_4+k_4)+\rho^3(u_{11}+k_{11}))+S(\rho(u_{14}^*+k_{14})+\rho^2(u_4^*+k_4)+\rho^3(u_{11}^*+k_{11})) \\ f_3' = S(\rho(u_9+k_9)+\rho^2(u_6+k_6)+\rho^3(u_{12}+k_{12}))+S(\rho(u_9^*+k_9)+\rho^2(u_6^*+k_6)+\rho^3(u_{12}^*+k_{12})) \end{cases}$$

So, we have the following relations,

$$\rho^2(f_1') = f_3' = \rho(f_2')$$

**Relations for random fault at $\Delta X_1^{r-1}$.** Similarly, for a random fault at $\Delta X_1^{r-1}$, a set

of relations can be generated. To distinguish the relations from the previous one, we define the $\Delta U^{r-1}$ as

$$\Delta U^{r-1} = \begin{pmatrix} 0 & p'_3 & 0 & 0 \\ p'_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & p'_2 & 0 \end{pmatrix}.$$

where

$$\begin{cases} p'_1 = S(\rho(u_5 + k_5) + \rho^2(u_{15} + k_{15}) + \rho^3(u_0 + k_0)) + S(\rho(u^*_5 + k_5) + \rho^2(u^*_{15} + k_{15}) + \rho^3(u^*_0 + k_0)) \\ p'_2 = S(\rho(u_6 + k_6) + \rho^2(u_{12} + k_{12}) + \rho^3(u_3 + k_3)) + S(\rho(u^*_6 + k_6) + \rho^2(u^*_{12} + k_{12}) + \rho^3(u^*_3 + k_3)) \\ p'_3 = S(\rho(u_1 + k_1) + \rho^2(u_{14} + k_{14}) + \rho^3(u_4 + k_4)) + S(\rho(u^*_1 + k_1) + \rho^2(u^*_{14} + k_{14}) + \rho^3(u^*_4 + k_4)) \end{cases}$$

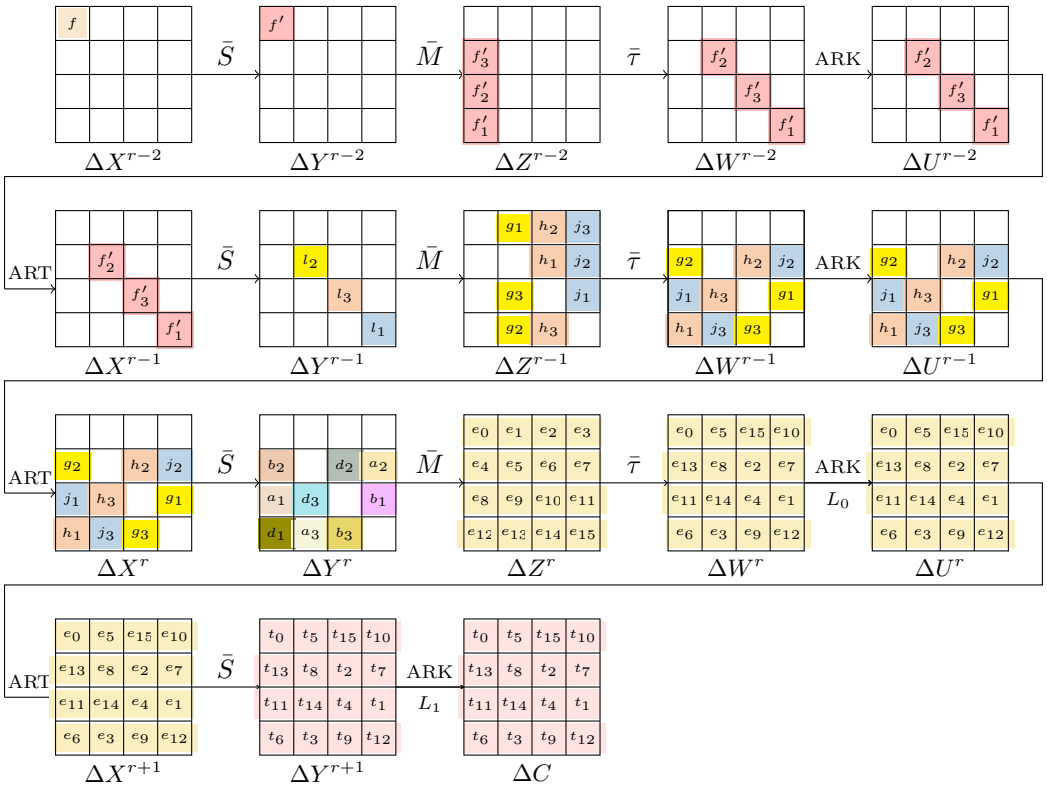



**Figure 6:** Fault propagation in QARMAv2-64 with fault induced at $0^{th}$ cell at the beginning of the $(r-2)$-th backward round. Each variable represents a non-zero fault value and an empty cell corresponds to a zero differential.

Again, we get the following relations,

$$\rho^2(p'_1) = p'_3 = \rho(p'_2)$$

**Relations for random fault at $\Delta X_0^{r-2}$.** To recover the key $L_0$, faults at only two different locations are not sufficient. We need some more faults. However, to minimize the number of faults, one more fault is induced at $\Delta X_0^{r-2}$ at the beginning of the $(r-2)$-th backward round. It is necessary here to mention that, again, the value of the fault is

random, and the cell location can also be random. For a fault at any cell in $\Delta X^{r-2}$, the same type of equations can be derived, and a similar process can be done as we will discuss here for the fault at $\Delta X_0^{r-2}$. We have chosen the random fault at $\Delta X_0^{r-2}$ only for the discussion purpose. In Fig. 6, we have presented the fault propagation of the random fault $f$ injected at $\Delta X_0^{r-2}$. Referring to Fig.6, as we know the key $L_1$, we know the original $U^r$ for a random plaintext and the faulty $U^{r*}$ for the same plaintext and the same key with the fault $\Delta U^r$. Again we define

$$U^r = \begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix}, \qquad U^{r*} = \begin{pmatrix} v_0^* & v_1^* & v_2^* & v_3^* \\ v_4^* & v_5^* & v_6^* & v_7^* \\ v_8^* & v_9^* & v_{10}^* & v_{11}^* \\ v_{12}^* & v_{13}^* & v_{14}^* & v_{15}^* \end{pmatrix}$$

where the $\Delta U^r$ is known to us and defined as in Fig. 6.

Now to recover the key $L_0$ we will concentrate on the grid $\Delta U^{r-1}$ in Fig. 6. The non-zero differentials in this grid are interrelated. Again we will use these relations to construct a set of equations in unknown key variables. Here the interrelation between the cells are given.

$$\Delta U^{r-1} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \Delta U_4^{r-1} & 0 & \Delta U_6^{r-1} & \Delta U_7^{r-1} \\ \Delta U_8^{r-1} & \Delta U_9^{r-1} & 0 & \Delta U_{11}^{r-1} \\ \Delta U_{12}^{r-1} & \Delta U_{13}^{r-1} & \Delta U_{14}^{r-1} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \rho^2(l_2) & 0 & \rho^2(l_3) & \rho^2(l_1) \\ \rho(l_1) & \rho^3(l_3) & 0 & \rho(l_2) \\ \rho(l_3) & \rho^3(l_1) & \rho^3(l_2) & 0 \end{pmatrix}.$$

where

$$\begin{cases} \Delta U_8^{r-1} = S(\rho(v_{15} + k_{15}) + \rho^2(v_0 + k_0) + \rho^3(v_{10} + k_{10})) + S(\rho(v_{15}^* + k_{15}) + \rho^2(v_0^* + k_0) + \rho^3(v_{10}^* + k_{10})) \\ \Delta U_7^{r-1} = S(\rho(v_8 + k_8) + \rho^2(v_2 + k_2) + \rho^3(v_{13} + k_{13})) + S(\rho(v_8^* + k_8) + \rho^2(v_2^* + k_2) + \rho^3(v_{13}^* + k_{13})) \\ \Delta U_{13}^{r-1} = S(\rho(v_{11} + k_{11}) + \rho^2(v_1 + k_1) + \rho^3(v_{14} + k_{14})) + S(\rho(v_{11}^* + k_{11}) + \rho^2(v_1^* + k_1) + \rho^3(v_{14}^* + k_{14})) \end{cases}$$

From the equations provided above, we can derive the following relation,

$$\rho^2(\Delta U_8^{r-1}) = \Delta U_{13}^{r-1} = \rho(\Delta U_7^{r-1})$$

Similarly, from $\Delta U^{r-1}$, we can write the other equations as follows,

$$\begin{cases} \Delta U_{11}^{r-1} = S(\rho(v_2 + k_2) + \rho^2(v_{13} + k_{13}) + \rho^3(v_7 + k_7)) + S(\rho(v_2^* + k_2) + \rho^2(v_{13}^* + k_{13}) + \rho^3(v_7^* + k_7)) \\ \Delta U_4^{r-1} = S(\rho(v_5 + k_5) + \rho^2(v_{15} + k_{15}) + \rho^3(v_0 + k_0)) + S(\rho(v_5^* + k_5) + \rho^2(v_{15}^* + v_{15}) + \rho^3(v_0^* + k_0)) \\ \Delta U_{14}^{r-1} = S(\rho(v_6 + k_6) + \rho^2(v_{12} + k_{12}) + \rho^3(v_3 + k_3)) + S(\rho(v_6^* + k_6) + \rho^2(v_{12}^* + k_{12}) + \rho^3(v_3^* + k_3)) \\ \Delta U_{12}^{r-1} = S(\rho(v_0 + k_0) + \rho^2(v_{10} + k_{10}) + \rho^3(v_5 + k_5)) + S(\rho(v_0^* + k_0) + \rho^2(v_{10}^* + k_{10}) + \rho^3(v_5^* + k_5)) \\ \Delta U_6^{r-1} = S(\rho(v_3 + k_3) + \rho^2(v_9 + k_9) + \rho^3(v_6 + k_6)) + S(\rho(v_3^* + k_3) + \rho^2(v_9^* + k_9) + \rho^3(v_6^* + k_6)) \\ \Delta U_9^{r-1} = S(\rho(v_4 + k_4) + \rho^2(v_{11} + k_{11}) + \rho^3(v_1 + k_1)) + S(\rho(v_4^* + k_4) + \rho^2(v_{11}^* + k_{11}) + \rho^3(v_1^* + k_1)) \end{cases}$$

Hence, by considering the above set of six equations mentioned earlier, we can extract the following connections, and by strategically selecting these relations, we can significantly reduce the key space for $L_0$.

$$\begin{cases} \rho^2(\Delta U_{11}^{r-1}) = \Delta U_{14}^{r-1} = \rho(\Delta U_4^{r-1}) \\ \rho^2(\Delta U_{12}^{r-1}) = \Delta U_9^{r-1} = \rho(\Delta U_6^{r-1}) \end{cases}$$

Now, let's discuss how to recover the key denoted as $L_0$ using the aforementioned relationships. First, let's examine the relationships: $\rho^2(p_1') = p_3'$, $\rho^2(\Delta U_8^{r-1}) = \Delta U_{13}^{r-1}$ &

$\rho^2(\Delta U_{12}^{r-1}) = \Delta U_9^{r-1}$. We have three independent equations involving eight unknown key variables: $k_0$, $k_1$, $k_4$, $k_5$, $k_{10}$, $k_{11}$, $k_{14}$ & $k_{15}$. By solving these equations, we can reduce our key space from approximately $2^{32}$ to $2^{20}$. Due to the non-bijective nature of S-Box for a given differential input, the number of solutions may not be precisely $2^{20}$, but it will be approximately $2^{20}$. By employing these same relationships with a different fault, we can similarly reduce the key space to around $2^{20}$.

Now consider the relations $\rho(f_1') = f_2'$, $\rho(\Delta U_8^{r-1}) = \Delta U_7^{r-1}$ & $\rho(\Delta U_{11}^{r-1}) = \Delta U_4^{r-1}$. Once again, we have three independent equations, but this time, they involve only four unknown key variables: $k_2$, $k_7$, $k_8$ & $k_{13}$. By solving these equations for two fault values, we can further reduce our key space $2^{48}$ to $2^{20} \times 2^4 = 2^{24}$ for the key variables $k_0$, $k_1$, $k_2$, $k_4$, $k_5$, $k_7$, $k_8$, $k_{10}$, $k_{11}$, $k_{13}$, $k_{14}$ & $k_{15}$.

Finally we have four more independent equations $\rho^2(f_1') = f_3', \rho(p_1') = p_2', \rho^2(\Delta U_{11}^{r-1}) = \Delta U_{14}^{r-1}$ & $\rho(\Delta U_{12}^{r-1}) = \Delta U_6^{r-1}$ with four unknown key variables $k_3$, $k_6$, $k_9$ & $k_{12}$. In a similar fashion, these four unknown key variables can be determined by solving the four equations. Furthermore, it's also worth noting that the key space remains consistent when considering the common keys that fulfill the four equations for two distinct fault values. After all these calculations, the key space for $L_0$ is reduced to $2^{24}$ from $2^{64}$ approximately. With the reduced key space, we can now perform a brute-force search. Therefore, both $L_0$ & $L_1$ can be recovered using six random faults at three distinct locations, two at the beginning of the $(r-1)$-th backward round and one at the beginning of $(r-2)$-th backward round.

*Remark* 2. The process of selecting the specific relations and fault locations for the recovery of the key $L_0$ is by no means a simple task. Our findings involve an exhaustive exploration of multiple alternatives, leading us to conclude that the aforementioned combination of relations and fault locations strikes the most optimal balance. This delicate balance is crucial for achieving the desired outcomes, considering the intricate interplay between complexity and the occurrence of minimal faults. Such a selection necessitated thorough analysis and critical decision-making, underscoring the nontrivial nature of our technique.

## 3.3 Filtering Appropriate Faulty Ciphertext

To carry out our key recovery attack, the first step is to filter the appropriate faulty ciphertexts. While we discussed this informally in section 3.2, here we describe the process more formally. The filtering method remains largely the same across all variants of `QARMAv2`. For clarity and illustration, we will explain it for `QARMAv2`-64. Suppose a random fault $f$ is injected at a specific location at the beginning of the $(r-1)$-th backward round. We can then use the fault propagation patterns shown in Fig. 4. Each state in this figure represents a unique difference pattern caused by a random fault $f$ after it passes through two inverse round functions $\bar{R}$ and an inverse S-box $\bar{S}$. This filtering step does not require knowledge of the secret key. As discussed in section 3.2.1, all the fault propagation patterns are unique. Therefore, if the attacker wants a single fault at a specific location at the beginning of the $(r-1)$-th backward round, she can look at the ciphertext difference. When the observed difference matches a known pattern, the corresponding ciphertext pair (original and faulty) can be used for key recovery. If the difference does not match any known pattern, it likely results from multiple faults, and such ciphertexts can be discarded.
As discussed in section 3.2.2, these unique patterns are enough to recover the key $L_1$. To recover $L_0$, the attacker needs to inject another random fault at the beginning of the $(r-2)$-th backward round. The same set of unique patterns from Fig. 4 can help in filtering valid faulty ciphertexts. In Fig. 6, the observed difference at $\Delta Y^r$ matches one of the known patterns from Fig. 4, which corresponds to a specific fault location. To identify the correct faulty ciphertext, we need to determine this $\Delta Y^r$, which can be done using already recovered key $L_1$. Given $L_1$ and the original and faulty ciphertexts ($C$ and $C^*$),

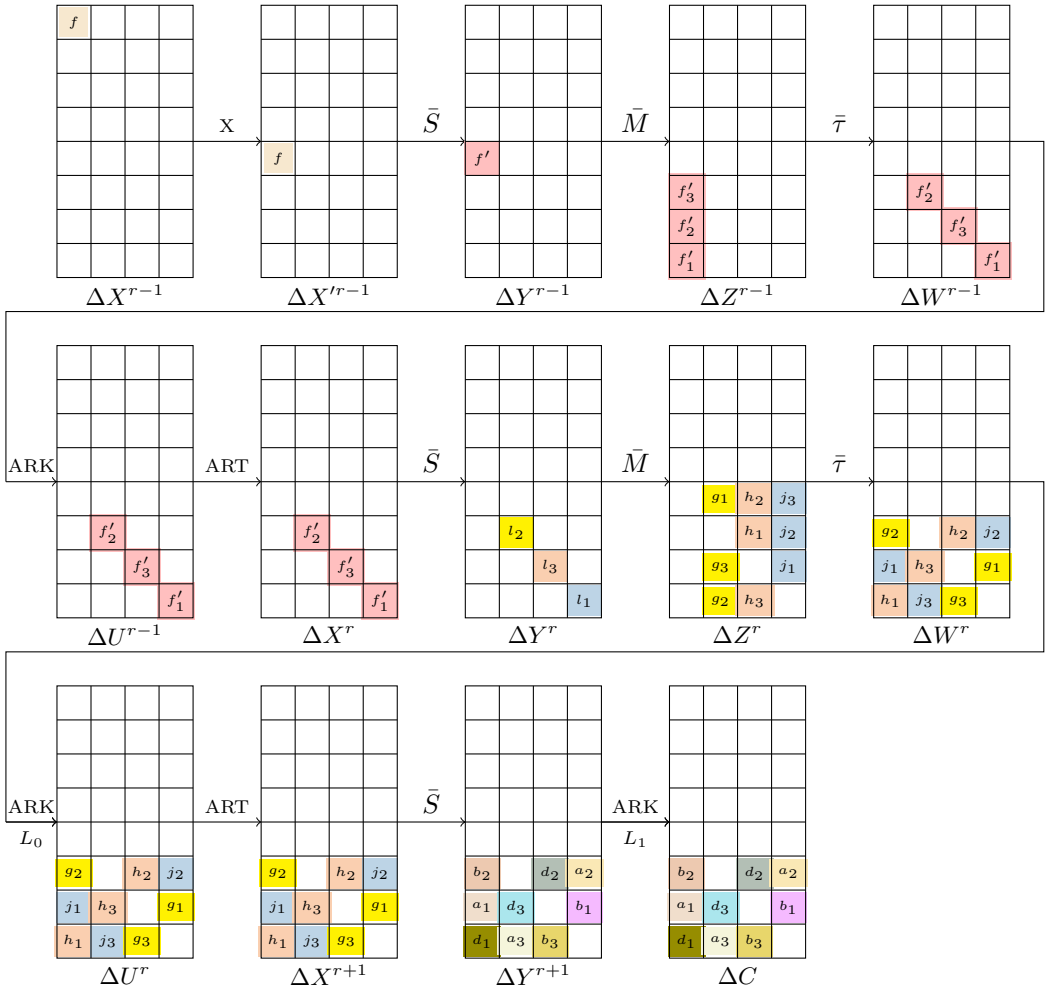



**Figure 7:** Fault propagation in QARMAv2-128 with fault induced at $0^{th}$ cell at the beginning of the $(r-1)$-th backward round where r is even. Each variable represents a non-zero fault value and an empty cell corresponds to a zero differential.

we compute $U^r$ and $U^{r*}$, and then the difference $\Delta U^r$. Since $\Delta W^r = \Delta U^r$, we can then compute $\Delta Y^r$ by applying the operations $M \circ \tau$ on $\Delta W^r$. Finally, we match the observed $\Delta Y^r$ with the known patterns in Fig. 4 to determine the fault location and thus select the correct faulty ciphertext. Once the valid faulty ciphertexts are identified, the key recovery attack can proceed.

## 3.4 Key Recovery Attack of QARMAv2-128-128

The key recovery attack of QARMAv2-128-128 is similar to the $L_1$ key recovery of QARMAv2-64. In QARMAv2-128-128, the same key is used in every round. Some interesting facts can be observed during the fault propagation. Similar to the QARMAv2-64, a random fault is injected at the beginning of the $(r-1)$-th backward round. However, the fault propagation differs for an even number of rounds and an odd number of rounds. For a random fault injection at $X^{r-1}$, if r is even, the `eXchangeRows` operation is done in the $(r-1)$-th backward round, but if r is odd, then the `eXchangeRows` operation is done in backward

round $r$. The fault propagations for even and odd rounds are presented in Fig. 7 and Fig. 8, respectively. One interesting fact is that we can choose a single fault cell location such that the faulty ciphertexts will be in a single layer, which is valid for both even and odd round cipher. This fact can be observed from Fig. 7 and 8. To recover the unique key $L_1$ of QARMAv2-128-128, when $r$ is even, we require two distinct fault locations from one layer and another two different fault locations from the other layer within $X^{r-1}$ and when $r$ is odd, we have the flexibility to select two fault locations from $X_0^{r-1}, X_5^{r-1}, X_{10}^{r-1}, X_{15}^{r-1}$ in the first layer and two additional fault locations from $X_{16}^{r-1}, X_{21}^{r-1}, X_{26}^{r-1}, X_{31}^{r-1}$ in the second layer of $X^{r-1}$. It's worth noting that there may be other viable choices as well.

Denote the original ciphertext text originating from random plaintext by $C$ and the faulty ciphertext by $C^*$ originating from the same plaintext using the same key $L_1$ with a random fault at a cell at the beginning of the $(r-1)$-th backward round i.e., in $X^{r-1}$.

$$C = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \\ c_{16} & c_{17} & c_{18} & c_{19} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{24} & c_{25} & c_{26} & c_{27} \\ c_{28} & c_{29} & c_{30} & c_{31} \end{pmatrix}, C^* = \begin{pmatrix} c_0^* & c_1^* & c_2^* & c_3^* \\ c_4^* & c_5^* & c_6^* & c_7^* \\ c_8^* & c_9^* & c_{10}^* & c_{11}^* \\ c_{12}^* & c_{13}^* & c_{14}^* & c_{15}^* \\ c_{16}^* & c_{17}^* & c_{18}^* & c_{19}^* \\ c_{20}^* & c_{21}^* & c_{22}^* & c_{23}^* \\ c_{24}^* & c_{25}^* & c_{26}^* & c_{27}^* \\ c_{28}^* & c_{29}^* & c_{30}^* & c_{31}^* \end{pmatrix}, L_1 = \begin{pmatrix} k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \\ k_{16} & k_{17} & k_{18} & k_{19} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{24} & k_{25} & k_{26} & k_{27} \\ k_{28} & k_{29} & k_{30} & k_{31} \end{pmatrix}.$$

When the round is even, we have assumed that the four different faults are induced at $X_0^{r-1}, X_1^{r-1}, X_{16}^{r-1}$ & $X_{17}^{r-1}$.

**Equations for fault injection at $X_0^{r-1}$.** The fault propagation of the random fault $f$ injected at $X_0^{r-1}$ is shown is Fig. 7 which is actually the similar propagation as in Fig 5. Similar to the $L_1$ key recovery of QARMAv2-64, we can construct a set of six independent equations as follows:

$$\begin{cases} \rho[S(c_{23}+k_{23})+S(c_{23}^*+k_{23})] &= S(c_{29}+k_{29})+S(c_{29}^*+k_{29}) \\ \rho^2[S(c_{24}+k_{24})+S(c_{24}^*+k_{24})] &= S(c_{29}+k_{29})+S(c_{29}^*+k_{29}) \\ \rho[S(c_{20}+k_{20})+S(c_{20}^*+k_{20})] &= S(c_{30}+k_{30})+S(c_{30}^*+k_{30}) \\ \rho^2[S(c_{27}+k_{27})+S(c_{27}^*+k_{27})] &= S(c_{30}+k_{30})+S(c_{30}^*+k_{30}) \\ \rho[S(c_{22}+k_{22})+S(c_{22}^*+k_{22})] &= S(c_{25}+k_{25})+S(c_{25}^*+k_{25}) \\ \rho^2[S(c_{28}+k_{28})+S(c_{28}^*+k_{28})] &= S(c_{25}+k_{25})+S(c_{25}^*+k_{25}) \end{cases} \tag{4}$$

As in the case of $L_1$ key recovery of QARMAv2-64, changing the fault value once, the keys $k_{20}$, $k_{22}$, $k_{23}$, $k_{24}$, $k_{25}$, $k_{27}$, $k_{28}$, $k_{29}$ & $k_{30}$ can be determined uniquely.

**Equations for fault injection at $X_1^{r-1}$.** Likewise, the introduction of random fault injections at $X_1^{r-1}$ assists us in deriving the following equations.

$$\begin{cases} \rho[S(c_{31}+k_{31})+S(c_{31}^*+k_{31})] &= S(c_{21}+k_{21})+S(c_{21}^*+k_{21}) \\ \rho^2[S(c_{16}+k_{16})+S(c_{16}^*+k_{16})] &= S(c_{21}+k_{21})+S(c_{21}^*+k_{21}) \\ \rho[S(c_{28}+k_{28})+S(c_{28}^*+k_{28})] &= S(c_{22}+k_{22})+S(c_{22}^*+k_{22}) \\ \rho^2[S(c_{19}+k_{19})+S(c_{19}^*+k_{19})] &= S(c_{22}+k_{22})+S(c_{22}^*+k_{22}) \\ \rho[S(c_{30}+k_{30})+S(c_{30}^*+k_{30})] &= S(c_{17}+k_{17})+S(c_{17}^*+k_{17}) \\ \rho^2[S(c_{20}+k_{20})+S(c_{20}^*+k_{20})] &= S(c_{17}+k_{17})+S(c_{17}^*+k_{17}) \end{cases} \tag{5}$$

We already know the unique key cells $k_{20}, k_{22}, k_{28}$ & $k_{30}$ from two random faults injection

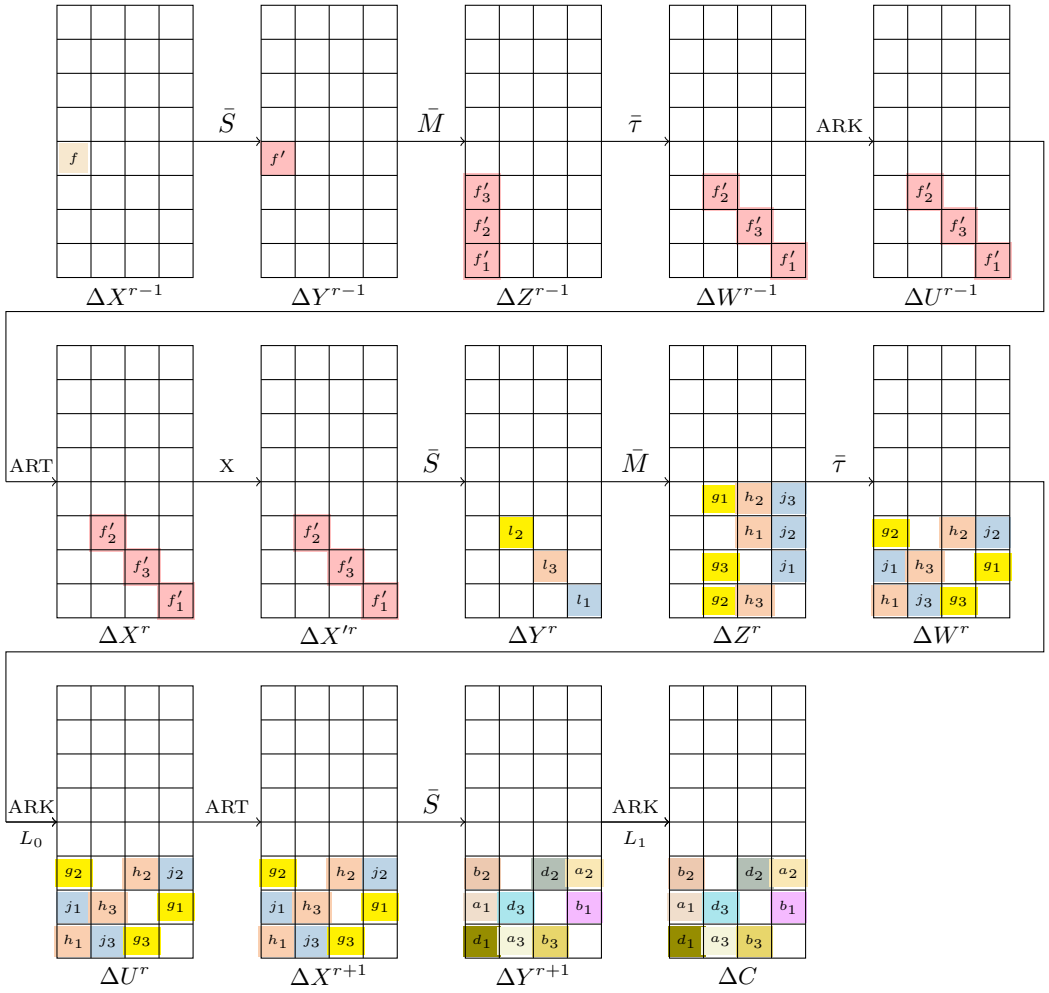



**Figure 8:** Fault propagation in QARMAv2-128 with fault induced at $16^{th}$ cell at the beginning of the $(r-1)$-th backward round where r is odd. Each variable represents a non-zero fault value and an empty cell corresponds to a zero differential.

at $X_0^{r-1}$. Utilizing the above system of equations i.e., Eqn. (5), the key space for the unknown key cells $k_{16}$, $k_{17}$, $k_{19}$, $k_{21}$ & $k_{31}$ is reduced to $2^4$ from $2^{20}$. Note that, here, we have not changed the fault value, i.e., only one fault is injected at the cell $X_1^{r-1}$ because the information of the known key cells makes the job simpler with an increased complexity of $2^4$. Combining the solutions for two faults injections at $X_0^{r-1}$ and one fault at $X_1^{r-1}$, we have reduced the key space for the 14 keys, $k_{16}$, $k_{17}$, $k_{19}$, $k_{20}$, $k_{21}$, $k_{22}$, $k_{23}$, $k_{24}$, $k_{25}$, $k_{27}$, $k_{28}$, $k_{29}$, $k_{30}$ & $k_{31}$ to $2^4$ from $2^{56}$.

**Equations for fault injection at $X_{16}^{r-1}$.** Similarly, a set of six independent equations can be generated due to the fault injection at $X_{16}^{r-1}$.

$$\begin{cases} \rho[S(c_7+k_7)+S(c_7^*+k_7)] &= S(c_{13}+k_{13})+S(c_{13}^*+k_{13}) \\ \rho^2[S(c_8+k_8)+S(c_8^*+k_8)] &= S(c_{13}+k_{13})+S(c_{13}^*+k_{13}) \\ \rho[S(c_4+k_4)+S(c_4^*+k_4)] &= S(c_{14}+k_{14})+S(c_{14}^*+k_{14}) \\ \rho^2[S(c_{11}+k_{11})+S(c_{11}^*+k_{11})] &= S(c_{14}+k_{14})+S(c_{14}^*+k_{14}) \\ \rho[S(c_6+k_6)+S(c_6^*+k_6)] &= S(c_9+k_9)+S(c_9^*+k_9) \\ \rho^2[S(c_{12}+k_{12})+S(c_{12}^*+k_{12})] &= S(c_9+k_9)+S(c_9^*+k_9) \end{cases} \tag{6}$$

Changing the fault value once we can recover the unique keys $k_4$, $k_6$, $k_7, k_8$, $k_9$, $k_{11}$, $k_{12}$, $k_{13}$ & $k_{14}$.

**Equations for fault injection at $X_{17}^{r-1}$.** Likewise, the following equations can be derived for introduction of a random fault injection at $X_{17}^{r-1}$.

$$\begin{cases} \rho[S(c_{15}+k_{15})+S(c_{15}^*+k_{15})] &= S(c_5+k_5)+S(c_5^*+k_5) \\ \rho^2[S(c_0+k_0)+S(c_0^*+k_0)] &= S(c_5+k_5)+S(c_5^*+k_5) \\ \rho[S(c_{12}+k_{12})+S(c_{12}^*+k_{12})] &= S(c_6+k_6)+S(c_6^*+k_6) \\ \rho^2[S(c_3+k_3)+S(c_3^*+k_3)] &= S(c_6+k_6)+S(c_6^*+k_6) \\ \rho[S(c_{14}+k_{14})+S(c_{14}^*+k_{14})] &= S(c_1+k_1)+S(c_1^*+k_1) \\ \rho^2[S(c_4+k_4)+S(c_4^*+k_4)] &= S(c_1+k_1)+S(c_1^*+k_1) \end{cases} \tag{7}$$

Similar to the case of fault injection at $X_0^{r-1}$ & $X_1^{r-1}$, injecting two random faults at $X_{16}^{r-1}$ & one random fault at $X_{17}^{r-1}$ and solving the corresponding generated equations, here also we can determine the key cells $k_0$, $k_1$, $k_3$, $k_4$, $k_5$, $k_6$, $k_7$, $k_8$, $k_9$, $k_{11}$, $k_{12}$, $k_{13}$, $k_{14}$ & $k_{15}$ with a complexity of $2^4$. So using six random faults at four different locations we have recovered 28 keys: $k_0$, $k_1$, $k_3$, $k_4$, $k_5$, $k_6$, $k_7$, $k_8$, $k_9$, $k_{11}$, $k_{12}$, $k_{13}$, $k_{14}$, $k_{15}$, $k_{16}$, $k_{17}$, $k_{19}$, $k_{20}$, $k_{21}$, $k_{22}$, $k_{23}$, $k_{24}$, $k_{25}$, $k_{27}$, $k_{28}$, $k_{29}$, $k_{30}$ & $k_{31}$ with $2^4 \times 2^4 = 2^8$ complexity. We are left with four keys $k_2, k_{10}, k_{18}$ & $k_{26}$. A simple brute force can be used to recover these four keys.

For odd $r$, similarly, we can construct four sets of equations for four different fault locations, of which two are injected in random but different locations of the first layer, and the other two are injected in random but different locations of the second layer. By solving these equations, we can recover the key $L_1$.

## 3.5 Key Recovery Attack of QARMAv2-128-192 and QARMAv2-128-256

The key recovery attacks on QARMAv2-128-192 and QARMAv2-128-256 are analogous to those on QARMAv2-64 and QARMAv2-128-128. The primary distinction lies in the number of fault injections increases as the key size grows. Specifically, the fault injection count for both QARMAv2-64 and QARMAv2-128-128 is six. In the case of QARMAv2-128-192 and QARMAv2-128-256, it becomes ten and twelve, respectively. We will first discuss how twelve random faults can recover the key $L_0$ and $L_1$ of QARAMAv2-128-256. Note that both keys are 128 bits, and both are independent. So, we need to recover both. The key recovery attack for QARMAv2-128-256 closely resembles the attack on QARMAv2-64. The key difference is that QARMAv2-128-256 comprises a block with two layers. Consequently, the number of faults required to recover the key of QARMAv2-128-256 is doubled. To recover the key, we must introduce four faults at two different locations of one layer and four faults at two different locations of another layer at the commencement of the $(r-1)$-th backward round. Additionally, we need to inject two faults at a location in one layer and two faults at a location in the other layer at the beginning of the $(r-2)$-th backward round.

In the context of QARMAv2-128-192, the process of recovering the key differs slightly compared to other variants. This divergence is primarily a result of variances in the key

schedule and the key stretching algorithm employed. The 192-bit master key, denoted as $K$, is structured as three 64-bit segments: $K = Y_0 || Y_1 || Y_2$ where $Y_i$ are 64-bit values. Then define $K_0 = Y_0 || Y_1$ and $K_1 = Y_2 || (\text{MAJ}(Y_0, Y_1, Y_2) \ggg 17)$ and the keys $L_0$ and $L_1$ are generated by key scheduling algorithm from using $K_0$ and $K_1$ respectively. Due to the key scheduling algorithm, in order to recover the master key $K$, it is necessary to retrieve both $L_0$ and $L_1$, and this can be done with twelve random faults injections as in the case of `QARMAv2-128-256`. However, to minimize the fault injection, we adopt an alternative approach.

### 3.5.1 Reducing Faults in `QARMAv2-128-192`

For `QARMAv2-128-192`, we will inject the fault in the decryption process instead of in the encryption process. This approach is ideal for our purposes. Referring to the Fig. 3 , if we inject eight faults at four different locations of $\tilde{X}^3$ , we can recover the key $K_0$, similar to the case of `QARMAv2-128-128` with the only difference is that here we recover the key $K_0$ instead of $L_1$. At this point, we have successfully retrieved the values of $Y_0$ and $Y_1$, and our remaining objective is to recover $Y_2$ rather than the key $K_1$. To achieve this, we only require two additional faults at a single location in the first layer of $\tilde{X}^4$. Hence, the master key $K = Y_0 || Y_1 || Y_2$ can be recovered with ten fault injections at five different locations. However, a challenge arises with the `eXchangeRows` operation. It becomes evident that when $r$ is even, the faults at $\tilde{X}^4_0$ are instrumental in recovering $Y_2$. This can be visualized by considering an additional layer at every step with no faults in any cell in Fig. 6. There are numerous other viable fault injection locations as well. Again, when $r$ is odd, the faults at $\tilde{X}^4_{16}$ can recover $Y_2$. It can also be visualized from Fig. 6 with the same assumption as for the case of even $r$ albeit with a minor adjustment in the initial state $\tilde{X}^4$. Here the fault is injected at $\tilde{X}^4_{16}$ instead of $\tilde{X}^4_0$. Nevertheless, after the first `eXchangeRows` operation, the fault will come to the $0^{th}$ cell. Again, various other favourable fault injection locations can be considered. Therefore, irrespective of whether $r$ is even or odd, the master key $K$ can be successfully recovered with just ten fault injections.

## 3.6 Discussion

In this study, we present a method for constructing a DFA attack on `QARMAv2-64`, `QARMAv2-128-128`, and extend our approach to target other variants of the `QARMAv2` cipher. Our method exploits the structural attributes of the diffusion matrix, requiring fault injection at the initiation of specific backward rounds to construct relations among the key bits, and non-zero differentials. Notably, we discover that the injection of two faults at distinct locations at the beginning of the $(r-1)$-th backward round facilitates the retrieval of information pertaining to 14 key cells of $L_1$ of `QARMAv2-64`. As far as our approach is concerned, this minimal number of faults, represents the most optimal solution we have identified.

Exploiting the characteristics of the S-box, we further demonstrate that the application of four faults enables the nearly unique recovery of the 14 cells of $L_1$, resulting in a reduction of the secret key space from $2^{64}$ to $2^8$. However, recognizing the considerable complexity associated with retrieving $L_1$ solely from these relations, we are prompted to explore additional patterns. Pursuing this line of motivation, we succeed in reducing the secret key space of $L_0$ from $2^{64}$ to $2^{24}$ by introducing only two faults at the beginning of the $(r-2)$-th backward round as we utilize the relations we derive after injecting four faults at two distinct locations at the beginning of $(r-1)$th backward rounds. It is noteworthy that the number of faults utilized in our approach is optimally minimal, ensuring a substantial reduction of the key space from $2^{128}$ to $2^{32}$ using this trick.

# 4 Fault Attack Complexity and Simulated Results

Differential Fault Analysis takes advantage of the interrelationship between the input and output differentials of the S-Box. Referring to the DDT table of QARMAv2 S-Box in Supplementary section, note that the maximum differential probability of S-Box is $2^{-2}$. That is, for fixed $\delta_i$ and $\delta_0$, the maximum number of $x$, satisfying the difference equation $SC(x + \delta_i) + SC(x) = \delta_o$ is $16 \times 2^{-2} = 4$. Define

$$Ns(\delta_i, \delta_o) = \#\{\ x \mid x \in \mathbb{F}_{2^4}, SC(x + \delta_i) + SC(x) = \delta_o\}$$

If $Ns(\delta_i, \delta_o)$ is not null, then it is equals to 2 with probability of 84.9% (90/106) for QARMAv2. This indicates that for a given fault difference equation, to recover the unique key, at least two faults are required at the same location, i.e., changing the fault value at least once we expect the unique key.

**Attack Complexity for QARMAv2-64.** In the case of $L_1$ key recovery of QARMAv2-64, solving the independent equations such as Eqn. (1), the key space for $k_7$, $k_8$ & $k_{13}$ reduces to approximately $2^4$ from $2^{12}$. Changing the fault at least once the keys $k_7$, $k_8$ & $k_{13}$ are determined uniquely with probability almost one. Similarly, from Eqn. (2), the keys $k_4$, $k_{11}$, $k_{14}$ and $k_6$, $k_9$, $k_{12}$ are recovered uniquely with probability almost one. So these nine keys can be recovered almost uniquely injecting two different random faults at $X_0^{r-1}$ at the start of the $(r-1)$-th backward round. In a similar way, injecting two different faults at $X_1^{r-1}$ at the start of the $(r-1)$-th backward round, we can recover the keys $k_0$, $k_1$, $k_3$, $k_4$, $k_5$, $k_6$, $k_{12}$, $k_{14}$, & $k_{15}$. In total, 14 out 16 keys are recovered almost uniquely. The remaining two keys can be recovered by a brute-force search. Therefore, using four faults at the start of the $(r-1)$-th backward round, the key $L_1$ of QARMAv2-64 is recovered with complexity $2^8$ approximately.

To recover the key $L_0$ of QARMAv2-64 two extra faults are injected at $X_0^{r-2}$ at the start of the $(r-2)$-th backward round. The relations $\rho^2(p'_1) = p'_3$, $\rho^2(\Delta U_8^{r-1}) = \Delta U_{13}^{r-1}$ & $\rho^2(\Delta U_{12}^{r-1}) = \Delta U_9^{r-1}$ give the key variables $k_0$, $k_1$, $k_4$, $k_5$, $k_{10}$, $k_{11}$, $k_{14}$ & $k_{15}$. Utilizing the same relations with the new fault, the key space reduces to $2^{20}$ from $2^{32}$. In a similar way the key variables $k_2$, $k_7$, $k_8$ & $k_{13}$ are recovered using the relations $\rho(f'_1) = f'_2$, $\rho(\Delta U_8^{r-1}) = \Delta U_7^{r-1}$ & $\rho(\Delta U_{11}^{r-1}) = \Delta U_4^{r-1}$. Therefore, the key variables $k_0$, $k_1$, $k_2$, $k_4$, $k_5$, $k_7$, $k_8$, $k_{10}$, $k_{11}$, $k_{13}$, $k_{14}$ & $k_{15}$ can be recovered with $2^{20} \times 2^4 = 2^{24}$ complexity. Finally, we can recover the remaining key variables $k_3$, $k_6$, $k_9$ & $k_{12}$ almost uniquely utilizing the four equations $\rho^2(f'_1) = f'_3$, $\rho(p'_1) = p'_2$, $\rho^2(\Delta U_{11}^{r-1}) = \Delta U_{14}^{r-1}$ & $\rho(\Delta U_{12}^{r-1}) = \Delta U_6^{r-1}$. Therefore, along with the four faults, injected to recover the key $L_1$, two more faults at the start of the $(r-2)$-th backward round recovers the key $L_0$ with complexity $2^{24}$ approximately. So, using four faults at the start of the $(r-1)$-th backward round and two faults at the start of the $(r-2)$-th backward round, the round keys $L_0$ and $L_1$ of QARMAv2-64 can be recovered with complexity $2^8 \times 2^{24} = 2^{32}$ approximately. So, the overall complexity is reduced to $2^{32}$ from $2^{128}$.

**Attack Complexity for QARMAv2-128-128.** In the case of QARMAv2-128-128, to recover the key $L_1$, we need six faults injections, two at $X_0^{r-1}$, one at $X_1^{r-1}$, two at $X_{16}^{r-1}$ and one at $X_{17}^{r-1}$. The keys $k_{16}$, $k_{17}$, $k_{19}$, $k_{20}$, $k_{21}$, $k_{22}$, $k_{23}$, $k_{24}$, $k_{25}$, $k_{27}$, $k_{28}$, $k_{29}$, $k_{30}$ & $k_{31}$ are recovered using the faults at $X_0^{r-1}$ and $X_1^{r-1}$ with a complexity of $2^4$. In a similar way, due to the faults injections at $X_{16}^{r-1}$ and $X_{17}^{r-1}$, the keys $k_0$, $k_1$, $k_3$, $k_4$, $k_5$, $k_6$, $k_7$, $k_8$, $k_9$, $k_{11}$, $k_{12}$, $k_{13}$, $k_{14}$ & $k_{15}$ are recovered with a complexity of $2^4$. The remaining four keys $k_2$, $k_{10}$, $k_{18}$ & $k_{26}$ can be recovered using a simple brute-force with a complexity of $2^{16}$. Therefore, the total complexity to recover the whole key $L_1$ for QARMAv2-128-128 is $2^4 \times 2^4 \times 2^{16} = 2^{24}$ approximately, reduced from $2^{128}$.

**Attack Complexity for QARMAv2-128-192 and QARMAv2-128-256.** The key recovery for QARMAv2-128-192 and QARMAv2-128-256 can be viewed as special cases of the recovery process for QARMAv2-64, as elaborated in section 3.5. In the case of QARMAv2-128-256, the recovery of keys requires twice as many faults compared to QARMAv2-64, specifically twelve faults. This key recovery process can be accomplished with a complexity of $2^{64}$ approximately. In the case of QARMAv2-128-192, we need ten faults of which eight at four different locations of $\tilde{X}^3$ and two at one location of $\tilde{X}^4$. The complexity of the key recovery attack is $2^{40}$ approximately. Please note that the key recovery attacks on QARMAv2-128-192 and QARMAv2-128-256 closely resemble the key recovery attack on QARMAv2-64. Therefore, it is not necessary to reiterate the same details. The complexity calculations for these attacks are also quite similar and can be readily confirmed.

**Simulation Results.** We conducted simulation experiments for both the fault injection process and the key recovery attack. As explained in section 3.3, we simulated the fault injection process. In this setup, an attacker injects a random nibble fault at a random location at the beginning of the $(r-1)$-th and $(r-2)$-th backward round in the QARMAv2 cipher. By observing the unique difference patterns shown in Fig. 4, the attacker can identify the exact fault location. This allows the attacker to filter out the appropriate faulty ciphertext and use it to build independent equations for the key recovery attack.

We have successfully executed key recovery attacks on QARMAv2-64 and QARMAv2-128-128 using a standard PC. Specifically, we achieved the unique recovery of the $L_1$ key of QARMAv2-64 by solving Eqn. 1, Eqn. 2, and Eqn. 3. This was accomplished with the introduction of two random faults at $X_0^{r-1}$ and an additional two random faults at $X_1^{r-1}$ at the start of the $(r-1)$-th backward round. While changing the fault value does not always guarantee a unique key, it does so most of the time. In some cases, two or three keys may be obtained. During the key recovery process of $L_0$ of QARMAv2-64, the key space unexpectedly reduces. For instance, solving the independent equations $\rho^2(p'_1) = p'_3$, $\rho^2(\Delta U_8^{r-1}) = \Delta U_{13}^{r-1}$, and $\rho^2(\Delta U_{12}^{r-1}) = \Delta U_9^{r-1}$, which involve eight unknown key variables such as $k_0$, $k_1$, $k_4$, $k_5$, $k_{10}$, $k_{11}$, $k_{14}$ & $k_{15}$, and changing the fault value once, reduces the key space from $2^{32}$ to around $2^{20}$. However, in our implementation, we observed that the key space reduces to approximately $2^{13}$, making the key recovery process easier. Additionally, we successfully recovered the full key of QARMAv2-128-128.

# 5 Conclusion and Future Directions

In this paper, we present a comprehensive analysis of the nibble-oriented fault propagation patterns within QARMAv2, a recently developed block cipher. Our focus lies in demonstrating the efficiency of the differential fault attack on the two variations of the cipher, QARMAv2-64 and QARMAv2-128-128, induced by random faults, a contribution that marks a first in the literature. Furthermore, we investigate the potential applicability of our attack approach to other variations of the cipher. Notably, we observe a noteworthy reduction in the secret key space, from $2^{128}$ to $2^{32}$ for QARMAv2-64 and from $2^{128}$ to $2^{24}$ for QARMAv2-128-128, leading to the feasible recovery of the secret key within a practical timeframe for both variants.

Moving forward, it is imperative to extend our research efforts beyond the current scope, delving into other potential fault attack methodologies such as meet-in-the-middle differential fault analysis, integral fault attack, and algebraic fault attack. These distinct approaches offer promising avenues for further enhancing the resilience of cryptographic systems, providing insights into potential vulnerabilities that might not be apparent through our current methodology.

## Acknowledgement

The authors are thankful to the reviewers of IEEE Transactions on Computers for their valuable comments and suggestions that greatly improved the quality of this paper. The first author is supported by the Prime Minister's Research Fellowship, Government of India.

## References

[1] Zahra Ahmadian, Akram Khalesi, Dounia M'foukh, Hossein Moghimi, and María Naya-Plasencia. Truncated differential cryptanalysis: new insights and application to qarmav1-n and qarmav2-64. *Des. Codes Cryptogr.*, 92(12):4549–4591, 2024.

[2] Subidh Ali and Debdeep Mukhopadhyay. Differential fault analysis of twofish. In Miroslaw Kutylowski and Moti Yung, editors, *Information Security and Cryptology - 8th International Conference, Inscrypt 2012, Beijing, China, November 28-30, 2012, Revised Selected Papers*, volume 7763 of *Lecture Notes in Computer Science*, pages 10–28. Springer, 2012.

[3] Subidh Ali, Debdeep Mukhopadhyay, and Michael Tunstall. Differential fault analysis of AES: towards reaching its limits. *J. Cryptogr. Eng.*, 3(2):73–97, 2013.

[4] 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.

[5] Roberto Avanzi, Subhadeep Banik, Orr Dunkelman, Maria Eichlseder, Shibam Ghosh, Marcel Nageler, and Francesco Regazzoni. The tweakable block cipher family qarmav2. *IACR Cryptol. ePrint Arch.*, page 929, 2023.

[6] Roberto Avanzi, Ionut Mihalcea, David Schall, Andreas Sandberg, and Héctor Montaner. Cryptographic protection of random access memory: How inconspicuous can hardening against the most powerful adversaries be? *IACR Cryptol. ePrint Arch.*, page 1472, 2022.

[7] Nasour Bagheri, Reza Ebrahimpour, and Navid Ghaedi. New differential fault analysis on PRESENT. *EURASIP J. Adv. Signal Process.*, 2013:145, 2013.

[8] Tim Beyne. An invariant of the round function of qarmav2-64. *IACR Cryptol. ePrint Arch.*, page 963, 2023.

[9] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[10] Johannes Blömer and Volker Krummel. Fault based collision attacks on AES. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography, Third International Workshop, FDTC 2006, Yokohama, Japan, October 10, 2006, Proceedings*, volume 4236 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2006.

[11] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

[12] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptol.*, 14(2):101–119, 2001.

[13] Wei Cheng, Yongbin Zhou, and Laurent Sauvage. Differential fault analysis on midori. In Kwok-Yan Lam, Chi-Hung Chi, and Sihan Qing, editors, *Information and Communications Security - 18th International Conference, ICICS 2016, Singapore, November 29 - December 2, 2016, Proceedings*, volume 9977 of *Lecture Notes in Computer Science*, pages 307–317. Springer, 2016.

[14] Christoph Dobraunig, Maria Eichlseder, Hannes Groß, Stefan Mangard, Florian Mendel, and Robert Primas. Statistical ineffective fault attacks on masked AES with fault countermeasures. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 315–342. Springer, 2018.

[15] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. Exploiting ineffective fault inductions on symmetric cryptography. *IACR Cryptol. ePrint Arch.*, page 71, 2018.

[16] Sho Endo, Yang Li, Naofumi Homma, Kazuo Sakiyama, Kazuo Ohta, Daisuke Fujimoto, Makoto Nagata, Toshihiro Katashita, Jean-Luc Danger, and Takafumi Aoki. A silicon-level countermeasure against fault sensitivity analysis and its evaluation. *IEEE Trans. Very Large Scale Integr. Syst.*, 23(8):1429–1438, 2015.

[17] Aymeric Genêt, Matthias J. Kannwischer, Hervé Pelletier, and Andrew McLauchlan. Practical fault injection attacks on SPHINCS. *IACR Cryptol. ePrint Arch.*, page 674, 2018.

[18] Hosein Hadipour and Yosuke Todo. Cryptanalysis of qarmav2. Cryptology ePrint Archive, Paper 2023/1833, 2023.

[19] Kitae Jeong, Yuseop Lee, Jaechul Sung, and Seokhie Hong. Improved differential fault analysis on PRESENT-80/128. *Int. J. Comput. Math.*, 90(12):2553–2563, 2013.

[20] Anup Kumar Kundu, Aikata, Banashri Karmakar, and Dhiman Saha. Fault analysis of the PRINCE family of lightweight ciphers. *J. Cryptogr. Eng.*, 12(4):475–494, 2022.

[21] Antoon Purnal, Victor Arribas, and Lauren De Meyer. Trade-offs in protecting keccak against combined side-channel and fault attacks. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 285–302. Springer, 2019.

[22] Sayandeep Saha, Arnab Bag, Debapriya Basu Roy, Sikhar Patranabis, and Debdeep Mukhopadhyay. Fault template attacks on block ciphers exploiting fault propagation. In Anne Canteaut and Yuval Ishai, editors, *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*, volume 12105 of *Lecture Notes in Computer Science*, pages 612–643. Springer, 2020.

[23] Ling Song and Lei Hu. Differential fault attack on the PRINCE block cipher. In Gildas Avoine and Orhun Kara, editors, *Lightweight Cryptography for Security and Privacy - Second International Workshop, LightSec 2013, Gebze, Turkey, May 6-7, 2013, Revised Selected Papers*, volume 8162 of *Lecture Notes in Computer Science*, pages 43–54. Springer, 2013.

[24] Junko Takahashi and Toshinori Fukunaga. Improved differential fault analysis on CLEFIA. In Luca Breveglieri, Shay Gueron, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2008, FDTC 2008, Washington, DC, USA, 10 August 2008*, pages 25–34. IEEE Computer Society, 2008.

[25] Qingyuan Yu, Xiaoyang Dong, Lingyue Qin, Yongze Kang, Keting Jia, Xiaoyun Wang, and Guoyan Zhang. Automatic search of meet-in-the-middle differential fault analysis on aes-like ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):1–31, 2023.

[26] Fan Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin, Wei He, Ruyi Ding, Samiya Qureshi, and Kui Ren. Persistent fault analysis on block ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):150–172, 2018.