The QARMAv2 Family of Tweakable Block Ciphers

(Extended Version)*

Roberto Avanzi[1,2], Subhadeep Banik[3], Orr Dunkelman[4], Maria Eichlseder[5], Shibam Ghosh[4], Marcel Nageler[5] and Francesco Regazzoni[6,3]

[1] Arm Germany GmbH, Grasbrunn, Germany
roberto.avanzi@arm.com
[2] Caesarea Rothschild Institute, University of Haifa, Haifa, Israel
roberto.avanzi@gmail.com
[3] Università della Svizzera Italiana, Lugano, Switzerland
subhadeep.banik@usi.ch, regazzoni@alari.ch
[4] Computer Science Department, University of Haifa, Haifa, Israel
orrd@cs.haifa.ac.il, sghosh03@campus.haifa.ac.il
[5] Graz University of Technology, Graz, Austria
maria.eichlseder@iaik.tugraz.at, marcel.nageler@iaik.tugraz.at
[6] University of Amsterdam, Amsterdam, The Netherlands
f.regazzoni@uva.nl

Abstract. We introduce the QARMAv2 family of tweakable block ciphers. It is a redesign of QARMA (from FSE 2017) to improve its security bounds and allow for longer tweaks, while keeping similar latency and area. The wider tweak input caters to both specific use cases and the design of modes of operation with higher security bounds. This is achieved through new key and tweak schedules, revised S-Box and linear layer choices, and a more comprehensive security analysis. QARMAv2 offers competitive latency and area in fully unrolled hardware implementations.
Some of our results may be of independent interest. These include: new MILP models of certain classes of diffusion matrices; the comparative analysis of a full reflection cipher against an iterative half-cipher; our boomerang attack framework; and an improved approach to doubling the width of a block cipher.

Keywords: Tweakable Block Ciphers · Lightweight Cryptography · Reflection Ciphers · Memory Encryption · Memory Integrity · Message Authentication Codes

*Besides the table of contents and lists of figures and tables, this version differs from the FSE/ToSC paper in the following ways: We have added two new appendices on the cryptanalysis of QARMA and on the properties of the diffusion matrices; The conclusions now include additional insights on the comparison between purely iterative ciphers and reflection ciphers; Some typographical errors have been rectified.

Contents

List of Figures iii

List of Tables iv

1 Introduction 1
1.1 Rationale for Short Blocks and Longer Tweaks . . . 1
1.2 Use Cases, Security Model, and Their Implications . . . 2
1.3 Results . . . 2
Outline of the Paper . . . 3

2 Definition of the Cipher 4
2.1 Cells, Blocks, Layers and the Internal State . . . 4
2.2 Permutations and Shuffles . . . 4
2.3 The Round Functions . . . 5
2.4 The S-Boxes . . . 6
2.5 The Diffusion Matrix . . . 6
2.6 The Reflector . . . 7
2.7 The Round Keys . . . 7
2.8 The Round Constants . . . 8
2.9 The Tweak Schedule . . . 8
2.10 The Complete Circuit . . . 9
2.11 Stretching Shorter Keys . . . 9
2.12 Stretching Single Block Tweaks . . . 9

3 Choice of the Building Blocks and Their Properties 9
3.1 The Orthomorphism . . . 9
3.2 The `StateShuffle` . . . 10
3.3 The Tweak Shuffles . . . 10
3.4 The S-Boxes . . . 14
3.5 Alternative Constructions . . . 15

4 Security Analysis 15
4.1 Differential Cryptanalysis . . . 16
4.2 Boomerang Attacks . . . 18
4.3 Linear Cryptanalysis . . . 19
4.4 Impossible-Differential and Zero-Correlation Cryptanalysis . . . 20
4.5 Integral Cryptanalysis and the Division Property . . . 20
4.6 Slide, Meet-in-the-Middle and other Structural Attacks . . . 21
4.7 Invariant Subspace Cryptanalysis and Non-Linear Invariants . . . 22
4.8 Algebraic Attacks . . . 23
4.9 Security Implications of the Reflector . . . 24
4.10 Comparison to `QARMAv1` . . . 24

5 Version for Pointer Authentication and Memory Integrity 24
5.1 Usage . . . 25
5.2 Security . . . 25

6 Hardware Implementation and Evaluation **26**
6.1 Methodology 26
6.2 Results 27

7 Conclusions and Open Questions **29**

Acknowledgements **30**

References **31**

Appendices **37**

A MILP Modeling **37**
A.1 Modeling `MixColumns` 37
A.2 Extending Solutions Inductively 39

B Tables of Trail Weights **39**

C A Remark on Minimal Weight Linear Trails **41**

D Characteristics for Differential, Impossible Differential, and Boomerang Attacks **41**

E Full Diffusion Property for the Two-Layer Version **41**

F Wide Block Versions **48**

G Cryptanalysis of `QARMAv1` **49**

H Properties of the Diffusion Matrices **49**

I Test Vectors **52**
I.1 `QARMAv2-64` 52
I.2 `QARMAv2-128` 52
I.3 `QARMAv2-64`-σ_0 52

List of Figures

1 Reflector structure of `QARMAv2`. 4
2 `QARMAv2` encryption for odd r. 6
3 Code for the round constant generating function PSI (Ψ). 8
4 Modeling State Transitions for Determining Good Tweak Shuffles. 12
a One active cell through two rounds. 12
b Rows and columns through τ. 12
5 Differential properties of the S-Box ק and of `MixColumns`. 16
a DDT of the 4-bit S-Box ק. 16
b BCT of the 4-bit S-Box ק. 16
c Cell-wise DDT of Class I Matrices. 16
6 Boomerang/sandwich distinguisher. 18

7 PMAC computed with a TBC when freshness is not provided. 25
8 PMAC computed with a TBC when freshness is provided. 25
9 Differential properties of `MixColumns`. 37
a `XOR` model. 37
b `QARMA` Class II. 37
c `QARMA` Class I. 37
10 Example of minimal weight iterative linear trail. 41
11 Sketch of key recovery attack on `QARMAv2` with $r = 6$ based on a differential distinguisher for $r = 4$. 42
12 Longest impossible differential distinguisher for `QARMAv2`. 43
13 Boomerang distinguisher for $7 + 5$ rounds of `QARMAv2-64`. 44
14 Boomerang distinguisher for $6 + 6$ rounds of `QARMAv2-64`. 45
15 Boomerang distinguisher for $10 + 2$ rounds of `QARMAv2-64`. 46
16 Selected trails from the proof of the four-round full diffusion property for $\ell = 2$. 47
17 The eight-bit S-Box. 48
18 The sixteen-bit S-Box. 48

List of Tables

1 Security claims and parameter choices. 15
a With two independent tweak blocks. 15
b With a single block tweak. 15
2 Estimated reach of various types of cryptanalysis. 16
3 Minimum number of active S-Boxes for related-tweak differential characteristics and linear characteristics for `QARMAv2` 17
a With two independent tweak blocks. 17
b With a single block tweak. 17
4 Bounds for sandwich characteristics for `QARMAv2-64` 19
a With two independent tweak blocks. 19
b With a single block tweak. 19
5 Comparative Evaluation Metrics for the STM 90nm Process and Library. . . . 28
6 Comparative Evaluation Metrics for the Nangate 15nm Process and Library. . 29
7 Comparative Evaluation Metrics for the TSMC 5nm Process and Library. . . . 30
8 The six shuffles τ_1 to τ_6. 39
9 Minimum number of active S-Boxes in related-tweak differential characteristics and in linear trails/non-related-value differential characteristics. 40
a Single layer cipher with two independent tweak blocks. 40
b Two layer cipher with two independent tweak blocks. 40
c Single layer cipher with a single block tweak. 40
d Two layer cipher with a single block tweak. 40
10 Lower bounds for the number of perturbed bits per perturbed cell in the composite 16-bit S-Box $\sigma_0^{\langle 4\rangle}$. 48
11 Cryptanalysis of `QARMAv1`. 49

1 Introduction

An ongoing trend within the industry revolves around implementing robust isolation between mutually untrusted processes on a shared computing device. Intel's SGX [Gue16] and TDX [Int21], AMD's SEV [KPW16], and Arm's CCA [MPS+21] offer access control mechanisms to safeguard program execution from attacks from hostile peer and higher-privileged software. Some of these solutions also protect against physical access threats through *cryptographic memory protection* like encryption and integrity checks [AMS+22]. This approach extends cryptographic memory hardening from smart cards, microcontrollers, and secure processors to address security needs in cloud computing, online gaming, and premium content distribution. Several 64-bit block ciphers have been proposed for this purpose, such as `PRINCE` [BCG+12], `MANTIS` [BJK+16], `QARMA` [Ava17], and `PRINCEv2` [BEK+20].

One aspect that sets `MANTIS` and `QARMA` apart is that they are *tweakable block ciphers* [LRW02] (TBC): In addition to the secret key and a text, they accept a *third* input, called the *tweak*. The tweak, together with the key, selects the permutation computed by the cipher, but, unlike the key, it is public and may even be under adversarial control. TBCs ease the design of modes of operation, and in fact one of their first applications has been to modes for memory encryption [HT13]. `QARMA` also exists in a version with a block size of 128 bits and keys up to 256 bits, to meet the needs of general-purpose computing. The small version of `QARMA` is also used to implement PAC, i.e. the *Pointer Authentication Code*, a control flow integrity mechanism on the Arm architecture [Arm16, Qua17].

`QARMAv2` (pronounced "karma vee two"), is a redesign of `QARMA` to allow for longer tweaks and tighter security margins. It exists in two general-purpose versions with block lengths of $b = 64$ and 128 bits, denoted by `QARMAv2`-b-s, where s is the bit size of the key (more correctly, the claimed security level in bits), as well as in a slightly lighter 64-bit block version to be used for the PAC or memory integrity. For $b = 128$, the design allows for key sizes of $s = 128$, 192 or 256 bits. For $b = 64$, s is always 128 and can be omitted.

This redesign is not a whim: *Since introducing `QARMA`, years of research and trial and error have provided a deeper understanding of the design of block ciphers and of the demands they face in real-world applications.* The first understanding led, for instance, to a different key schedule, and to a rethinking of the relation between the substitution and diffusion layers. The second one led to the implementation of longer tweaks: We explain why these are needed in Section 1.1 but, before proceeding, *we note that throughout the remainder of the paper the older cipher `QARMA` will be called `QARMAv1` to avoid ambiguity.*

1.1 Rationale for Short Blocks and Longer Tweaks

Let us start by looking at `AES` in XTS mode [IEE18]. This mode is designed around the XEX construction [Rog04]. It allows the use of a 128-bit nonce, which is encrypted to obtain a first *mask*. The mask is added to plaintext and ciphertext, and then updated via an LFSR for each block. A longer nonce must be first hashed to 128 bits. Hence, care is due in order to prevent tweak reuse. Even worse, a counter mode like the GCM [MV04] uses a 96-bit IV. In these constructions, the block size of the underlying cipher bounds the space of the *permutations* that, for a fixed key, are parameterized by the tweak.

Ciphers with 256-bit blocks enable the use of longer uncompressed IVs, whether they are non-tweakable ciphers incorporated within the aforementioned constructs or bespoke designs, as long as they support 256-bit tweaks. Nevertheless, opting for a reduced block size facilitates the creation of designs characterized by smaller area, latency, and energy usage. Thus, a 128-bit TBC with 256-bit "native" tweaks offers the freedom to employ random or synthetic nonces/IVs without the concern of encountering repeated permutations. The sole consideration is that in numerous modes of operation, the necessity for re-keying or re-tweaking arises well before processing 2^{64} blocks, a concern that a 256-bit block cipher effectively sidesteps in real-world scenarios. From our perspective, this seems like a small price to pay for the ensuing area and performance savings.

Future-proofing memory encryption also supports 256-bit tweaks: Already with 48- to 52-bit addresses, 56-bit counters, and 32-bit process domain IDs in the tweak, more than 128 bits are necessary. For `QARMAv2`-64, 128-bit tweaks should suffice for most embedded applications. For PAC, 128-bit tweaks are required for improved versions of the feature.

Remark 1. The NIST, announcing their decision to update SP 800-38A [NIS01], has expressed interest in standardizing a "tweakable wide encryption technique" [NIS23]. In this context, wide TBCs have been considered [KLMR16, BLLS22]. We believe that a 128-bit block cipher with 256-bit keys and tweaks (yielding a 512-bit permutation space), combined with a two-pass mode of operation represents a strong proposition. However, our design principles can accommodate even wider designs if needed, as illustrated in Appendix F.

1.2 Use Cases, Security Model, and Their Implications

The goal of the `QARMAv2` design is to provide a general-purpose TBC that is also ideally suitable to cryptographic memory protection, as exemplified in [AMS+22], and fast computation of short-message MACs, for instance for control flow integrity. The design allows efficient hardware implementations, but at the same time programming optimized software implementations should be straightforward.

Several constructions of TBCs from non-tweakable block ciphers exist [Rog04, LRW11, LST12, Men15, JLM+17, Men18, JN20], but they all vastly increase latency. A different school of thought modifies the design of the cipher itself, such as the `TWEAKEY` framework [JNP14] or *TNT* [BGGS20], achieving lower overheads.

Our approach is related to `TWEAKEY`. We note that the latter unifies key and tweak, treating them as a single undifferentiated input. This approach may not reflect the very different real-world security requirements on keys and tweaks. In a system that uses a TBC, the key is changed infrequently, relatively few keys are used, and they can be generated by a physically hardened circuit without impacting overall performance. So, it is less susceptible to manipulation or related-key attacks. Conversely, the tweak changes often, it is public and the adversary may even be capable to choose its value, forcing consideration of related-tweak attacks. With a unified schedule, the cryptanalysis may overestimate the number of rounds required to reach a target security level. (On the other hand, as a result, `TWEAKEY` allows flexibility to adjust the relative sizes of key and tweak.)

These considerations prompt us to keep separate key and tweak schedules, as in `QARMAv1`. The simple, alternating key schedule employs non-sparse round constants and orthomorphisms as a first line of defense against structural attacks. Designing the tweak schedule is the main challenge: it must be designed in a way that does not adversely impact security while keeping the total area and latency of the cipher contained.

1.3 Results

`QARMAv2` is a reflector construction, a design spearheaded by `PRINCE` where an iterative sub-cipher is composed with its inverse, and a small involutory function is placed in between. This allows the use of a single algorithm for both encryption and decryption. The data path is kept as similar as possible to `QARMAv1`'s, a structure which has been proven so far very resistant to cryptanalysis. Still, the new design provides significantly improved security levels with respect to `QARMAv1`, as well as better suitability to practical applications and design of modes of operation due to the longer tweaks.

This is achieved through the introduction of new key and tweak schedules, a re-definition of the 128-bit version, a deeper analysis of the properties of the building blocks, and more thorough cryptanalysis in general. Just as `QARMAv1` can reuse part of the cryptanalysis of `MIDORI` and `MANTIS`, `QARMAv2` reuses parts of the cryptanalysis of `QARMAv1`.

While not consistently lighter than QARMAv1 at comparable security levels, we feel that with QARMAv2 we have improved the usefulness of the design since we increased the tweak length, while not significantly impacting performance.

The following results may have independent value:

1. We analyze both half-cipher and full-cipher and therefore we can assess the impact of the reflector construction;

2. The new MILP model for our Almost-MDS matrix can prove useful also in the study of other ciphers that use a similar matrix;

3. We propose novel MILP models for various attacks, sometimes marking a first to reflector constructions, for instance for boomerang distinguishers; and

4. We adopt a technique for doubling the width of a block cipher that involves parallel operation of two instances of the data obfuscation path of the cipher, which we call *layers*, with occasional cell swaps between them. While similar to the constructions of AESQ [KLMR16] and Pholkos [BLLS22], two important aspects of our design are:

   - If the key and tweak schedules and the cell swap operation are suitably defined, this approach can *directly* leverage existing cryptanalysis of the base design – for instance cell-wise models of the smaller design apply directly to the wider design. However, some aspects like full diffusion still require separate consideration.
   - We do not shuffle between the layers after each round like in AESQ, but still *before* full diffusion has occurred, unlike Pholkos. Thus, with a suitable tweak schedule we can reduce early self-cancellations arising from the tweak addition.

QARMAv2, just as its predecessor, offers significantly better energy-per-bit values than most ciphers in fully unrolled implementations, including the AES. This makes it ideal for always-on applications such as memory encryption or high-bandwidth network links. The cipher is flexible enough to be deployed for general purpose usage. Finally, we also allow the use of 128- and 192-bit keys for QARMAv2-128, besides 256-bit keys. This has been done to align with the AES and to allow deployers to opt for further latency and area savings while still keeping a solid security level. We expect the quantum security of QARMAv2 to be equivalent to similar ciphers with block and key of 128 bits, e.g. AES-128 [JNRV20].

We anticipate that QARMAv2 will be deployed in various settings such as for Pointer Authentication and for cryptographic memory protection [AMS+22].

Outline of the Paper. In Section 2, we present the specification of QARMAv2. In Section 3, we discuss how the components have been chosen. In Section 4, we evaluate the resistance of the design to various cryptanalytic approaches. In Section 5, we discuss the versions of QARMAv2 for PAC and memory integrity, and their security. In Section 6, we provide hardware implementation results. In Section 7, we conclude and state open questions.

In the appendices, we describe our MILP models of the diffusion matrices (Appendix A), discuss selected trails and characteristics occurring in our cryptanalysis (Appendices B to D), and prove the four-round full diffusion property for the large version of QARMAv2 (Appendix E). We also discuss the possibility of wider variants of the cipher (Appendix F), summarize the published cryptanalysis on QARMAv1 (Appendix G) as well as the properties of the diffusion matrices used in QARMAv1 and QARMAv2 (Appendix H), before finally providing test vectors for the various versions of QARMAv2 (Appendix I).

2 Definition of the Cipher

QARMAv2 follows the *reflector construction* of PRINCE, MANTIS, and QARMAv1, cf. Figure 1:

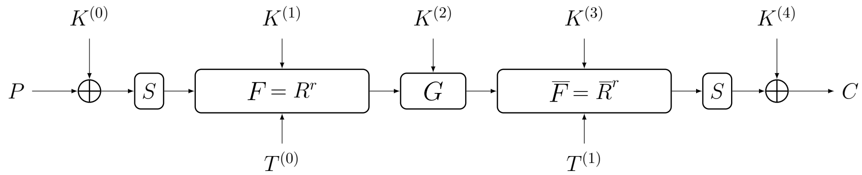


Figure 1: Reflector structure of QARMAv2.

The design is the composition of a *forward function*, of a symmetric *reflector*, also called *central construction*, and of a *backward function* which is the functional inverse of the forward function. This allows to use the same circuit for both encryption and decryption with a relatively minor set-up step. In Figure 1, we represent the initial and final rounds separately: They consist of just a key addition and a substitution layer, and are not tweaked. The reflector is also not tweaked. The values $K^{(i)}$, resp. $T^{(i)}$ are derived from the key K, resp. tweak T by simple operations. The function F is a keyed and tweaked iterated cipher with round function R. Here and elsewhere in the paper, a bar over a function denotes its inverse, for instance $\overline{R} = R^{-1}$.

The operator "+" on keys, tweaks, and states shall always denote the binary XOR.

We present the cipher as pseudo-code in Algorithm 1 and graphically in Figure 2.

The remaining notation is explained next.

2.1 Cells, Blocks, Layers and the Internal State

In Algorithm 1, $\mathscr{S}$ is the internal state of the cipher. Its size is b bits. A b-bit value is called a *block* and is represented as a three-dimensional array, consisting of ℓ *layers*, with $\ell \in \{1, 2\}$. A layer is an array of 16 elements, and also a 4 by 4 matrix of 4-bit *cells*:

$$\mathscr{L} = c_0 \| c_1 \| \cdots \| c_{14} \| c_{15} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{pmatrix} .$$

Thus, $b = 64\,\ell$. With respect to cell numbering, the b bits of a block are indexed in big endian order: Hence, bits $[b-1 \mathinner{..} b-4]$ are contained in the first cell of layer number 0, and bits $[3 \mathinner{..} 0]$ are in the fifteenth cell of the last layer. The bits in a cell are indexed in little endian order. The data obfuscation path is one block wide.

2.2 Permutations and Shuffles

A permutation π on $[0 \mathinner{..} 15]$ acts on a layer as follows:

$$(\pi(\mathscr{L}))_i = c_{\pi(i)} \quad \text{for} \quad 0 \le i < 16 \ . \tag{1}$$

Our choice for the *state shuffle* τ is MIDORI's shuffle

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

Algorithm 1: The `QARMAv2` Algorithm.

$\texttt{QARMAv2}_r\ \big[\mathsf{op},\ K_0,\ K_1,\ W_0,\ W_1,\ T_0,\ T_1,\ P\big] \mapsto C$

$t_0 \mapsfrom T_0$, $t_1 \mapsfrom T_1$ (Round tweak setup)
$\mathcal{k}_0 \mapsfrom K_0$, $\mathcal{k}_1 \mapsfrom K_1$ (Round key setup)
$\mathscr{S} \mapsfrom P + \mathcal{k}_0$ (Round $\#0$)
$\mathscr{S} \mapsfrom S(\mathscr{S})$
for $i = 1$ to r do (Rounds from $\#1$ to $\#r$)
  $\mathscr{S} \mapsfrom \mathscr{S} + \mathcal{k}_{i \bmod 2} + t_{i \bmod 2} + \mathfrak{c}_i$
  $\mathscr{S} \mapsfrom (S \circ M \circ \tau)(\mathscr{S})$
  if $i \equiv 1 \bmod 2$ then $t_1 \mapsfrom \varphi(t_1)$ else $t_0 \mapsfrom \bar{\varphi}(t_0)$
  $\{$ if $i \equiv r \bmod 2$ then $\mathscr{S} \mapsfrom \texttt{eXchangeRows}(\mathscr{S})$ $\}$ (Only for $\ell = 2$)
$\mathcal{k}_0 \mapsfrom o(\mathcal{k}_0)$, $\mathcal{k}_1 \mapsfrom \bar{o}(\mathcal{k}_1)$ (Round key transform)
if op $=$ enc then (Encryption)
  $\mathcal{k}_0 \mapsfrom \mathcal{k}_0 + \alpha$, $\mathcal{k}_1 \mapsfrom \mathcal{k}_1 + \beta$
else (Decryption)
  $\mathcal{k}_0 \mapsfrom \mathcal{k}_0 + o(\beta)$, $\mathcal{k}_1 \mapsfrom \mathcal{k}_1 + o^{-1}(\alpha)$
$\mathscr{S} \mapsfrom \tau(\mathscr{S})$ (Reflector)
$\mathscr{S} \mapsfrom M \cdot (\mathscr{S} + W_{r+1 \bmod 2}) + W_{r \bmod 2}$
$\mathscr{S} \mapsfrom \bar{\tau}(\mathscr{S})$
for $i = r$ down to 1 do (Rounds from $\#r+1$ to $\#2r$)
  $\{$ if $i \equiv r \bmod 2$ then $\mathscr{S} \mapsfrom \texttt{eXchangeRows}(\mathscr{S})$ $\}$ (Only for $\ell = 2$)
  $\mathscr{S} \mapsfrom (\bar{\tau} \circ \overline{M} \circ \overline{S})(\mathscr{S})$
  $\mathscr{S} \mapsfrom \mathscr{S} + \mathcal{k}_{i+1 \bmod 2} + t_{i+1 \bmod 2} + \mathfrak{c}_i$
  if $i > 1$ and $i \equiv 0 \bmod 2$ then $t_1 \mapsfrom \varphi(t_1)$ else $t_0 \mapsfrom \bar{\varphi}(t_0)$
$\mathscr{S} \mapsfrom \overline{S}(\mathscr{S})$ (Round $\#2r+1$)
$C \mapsfrom \mathscr{S} + \mathcal{k}_1$
return C

i.e. it acts on each layer as follows

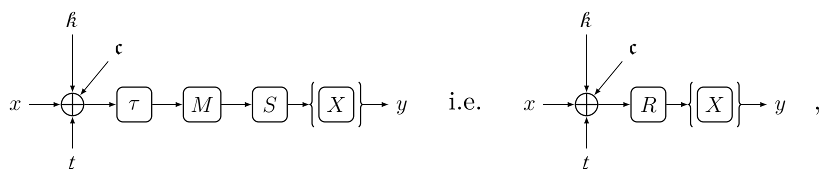
$$\mathscr{L} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{pmatrix} \stackrel{\tau}{\longmapsto} \begin{pmatrix} c_0 & c_{11} & c_6 & c_{13} \\ c_{10} & c_1 & c_{12} & c_7 \\ c_5 & c_{14} & c_3 & c_8 \\ c_{15} & c_4 & c_9 & c_2 \end{pmatrix} = \tau(\mathscr{L}) \ .$$

For $\ell = 2$, the action of a permutation on $[0\,..\,31]$ is similarly defined using a 8 by 4 matrix, wherein a permutation on $[0\,..\,15]$ acts on the two layers in a block in parallel, i.e. $\pi(i+16) := \pi(i) + 16$ for $0 \le i < 16$, and thus $c_{\pi(i+16)} = c_{\pi(i)+16}$.

2.3 The Round Functions

`QARMAv2` reuses the `QARMAv1` 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:



(3)

where $R = S \circ M \circ \tau$. The notations $\mathcal{k}$, resp. t, $\mathfrak{c}$ denote a round key, resp. tweak and constant: The `XOR` of these values at the same round is the latter's *round tweakey*. τ is

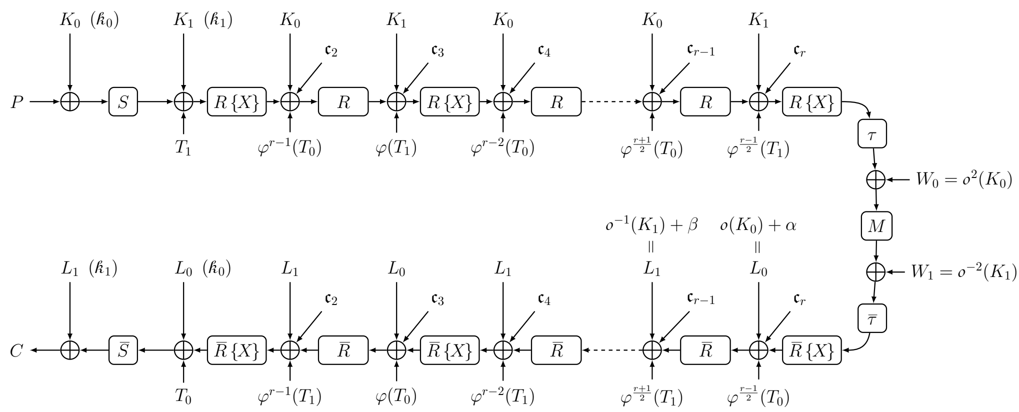


Figure 2: `QARMAv2` encryption for odd r. If r is even, W_0 and W_1 are swapped and the forward function starts with R instead of $R\,\{X\}$.

the `MIDORI StateShuffle` operation (2). The 4 by 4 matrix M operates column-wise by left multiplication on each layer of a block. S is the parallel application of $16\,\ell$ identical S-Boxes to all cells of the state. All these components are defined later in this section.

X denotes the `eXchangeRows` operation, that applies only for $\ell = 2$. It swaps the first two rows between the two layers. It is performed every other full round, where two `eXchangeRows`'s always flank the reflector, or, in other words, `eXchangeRows` is always included in Rounds r and $r+1$. X and S clearly commute with each other.

The half round function consists of just a round key addition and a S-Box layer and is only used for the first and the last rounds of the cipher.

2.4 The S-Boxes

For the general-purpose versions of `QARMAv2`, we use the following S-Box

$$ק = [\ 4\ \ 7\ \ 9\ \ \mathrm{B}\ \ \mathrm{C}\ \ 6\ \ \mathrm{E}\ \ \mathrm{F}\ \ 0\ \ 5\ \ 1\ \ \mathrm{D}\ \ 8\ \ 3\ \ 2\ \ \mathrm{A}\ ]\ . \tag{4}$$

For the PAC and memory authentication applications we optionally allow the use of `QARMAv1`'s σ_0

$$\sigma_0 = [\ 0\ \ \mathrm{E}\ \ 2\ \ \mathrm{A}\ \ 9\ \ \mathrm{F}\ \ 8\ \ \mathrm{B}\ \ 6\ \ 4\ \ 3\ \ 7\ \ \mathrm{D}\ \ \mathrm{C}\ \ 1\ \ 5\ ]\ . \tag{5}$$

2.5 The Diffusion Matrix

Let ρ denote the cyclic rotation to the left of the four bits in a cell, i.e. $\rho((x_3, x_2, x_1, x_0)) = (x_2, x_1, x_0, x_3)$. It is a linear transformation of $\mathbb{F}_2^4$, and $\rho^4 = 1$, the identity map. The diffusion matrix M is defined as the circulant matrix

$$M := M_{4,1} = \mathrm{circ}(0, \rho, \rho^2, \rho^3) = \begin{pmatrix} 0 & \rho & \rho^2 & \rho^3 \\ \rho^3 & 0 & \rho & \rho^2 \\ \rho^2 & \rho^3 & 0 & \rho \\ \rho & \rho^2 & \rho^3 & 0 \end{pmatrix}\ . \tag{6}$$

For the properties of these matrices and their classification we refer to Appendix H. Following the `QARMAv1` paper, this and other diffusion matrices, such as the `MIDORI` circulant $M_0 := \mathrm{circ}(0,1,1,1)$ and $M_{4,2} = \mathrm{circ}(0, \rho, \rho^2, \rho)$, are Almost-MDS (i.e. they have differential branch number equal to 4) and are grouped into *classes* depending on their transition patterns: *Class I* includes M_0 and $M_{4,1}$; and $M_{4,2}$ is a *Class II* matrix. Their transition patterns are displayed in Figure 9 in the Appendix, next to a simpler `XOR` model. Class I matrices have a slightly better diffusion model, but in `QARMAv1` Class II matrix $M_{4,2}$ was chosen on the basis of certain heuristics. For `QARMAv2` we use Class I matrix $M_{4,1}$ instead, validating this choice with a deeper cryptanalysis.

2.6 The Reflector

The reflector (or central construction) is similar to that of `QARMAv1`:

$$x \longrightarrow \boxed{\tau} \longrightarrow \overset{k_0 \downarrow}{\oplus} \longrightarrow \boxed{M} \longrightarrow \overset{k_1 \downarrow}{\oplus} \longrightarrow \boxed{\overline{\tau}} \longrightarrow y \quad , \tag{7}$$

where k_0, k_1 are two round keys. This function can be implemented using the original `QARMAv1` central construction, by using the round key $M \cdot k_0 + k_1$. Since the reflector does not contain S-Box layers, it does not count as a round.

2.7 The Round Keys

Let $o : \mathbb{F}_2^b \to \mathbb{F}_2^b$ be the orthomorphism

$$o(w) := (w \ggg 1) + (w \gg (b-1)) \ . \tag{8}$$

The key schedule, for the forward and backward functions, can be implemented by two registers k_0 and k_1, which are alternatingly added to the state. For encryption, they are initialized with the values K_0 and K_1, the two halves of the key $K = K_0 \| K_1$. The two round keys added in the reflector are $W_0 = o^2(K_0)$ and $W_1 = o^{-2}(K_1)$. During the computation of the reflector, K_0 and K_1 are subject to the transformation

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

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

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

Thus, for decryption we start with $k_0 \hookleftarrow L_1$ and $k_1 \hookleftarrow L_0$ and apply ι_d instead of ι_e, which differ only in the constants. In hardware, the latter can be multiplexed.

Remark 2. The circuit would be simpler if $\beta = o^{-1}(\alpha)$. However, in this case the untweaked backward function would be the inverse of the untweaked forward function for a relatively small class of weak keys, namely those with $K_1 = o(K_0) + \alpha$.

The following attack would apply: The reflector has $2^{48\ell}$ fixed points, so for these values the function would be the identity. Then, an adversary encrypts $\approx 2^{16\ell}$ plaintexts before it finds one with equal ciphertext. On the other hand, if the key is not in the weak set, assuming that the cipher behaves like a random permutation, the adversary would have to try the whole codebook. A weak key is finally brute-forced in time $2^{64\ell}$.

Taking the tweak schedule into account (cf. Section 2.9), the above attack works if the adversary sets $T_0 = T_1$. Therefore, we choose $\beta \neq o^{-1}(\alpha)$. This means that, similarly to `PRINCEv2` [BEK+20], `QARMAv2` does not enjoy the simple α-reflection property.

2.8 The Round Constants

The round constants are chosen to be of density approximately 1/2 and to represent all 4-bit values as homogeneously as possible, within the constraint of being generated programmatically. To facilitate lightweight SW and round-based HW implementations, the round constants are generated by a Galois LFSR seeded by the first and second 16 digits of the hexadecimal expansions of the fractional parts of π.

The 64 most significant bits of the round constant $\mathfrak{c}_2$ (the first non-zero round constant) are given by `0x243F6A8885A308D3ULL`, and each successive value is obtained from the LFSR's state after applying the LFSR 23 times. We call this update function Ψ. 128-bit values are generated by concatenating two consecutive outputs of Ψ, the first one giving the most significant bits. Round constants are given in big endian order.

The 64 most significant bits of α are given by $\alpha_0 =$ `0x13198A2E03707344ULL`. The 64 least significant bits if α (for $\ell = 2$) and β are generated from this seed in the same way as the $\mathfrak{c}$'s, i.e. $\beta := \Psi(\alpha)$ for $\ell = 1$, and $\alpha = \alpha_0 \| \Psi(\alpha_0)$ $\beta = \Psi^2(\alpha_0) \| \Psi^3(\alpha_0)$ for $\ell = 2$.

The LFSR is defined by the primitive polynomial $X^{64} + X^{50} + X^{33} + X^{19} + 1$ over $\mathbb{F}_2$. The polynomial was chosen to facilitate software implementations of Ψ, even on 16-bit microcontrollers. Code is given in Figure 3.

```
uint64_t PSI(uint64_t in)          // Galois LFSR - ticked 23 (13+10) times.
{
   uint64_t spill, tmp;
   spill = in >> 51;
   tmp =  (in << 13) ^ (spill << 50) ^ (spill << 33) ^ (spill << 19) ^ spill;
   spill = tmp >> 54;
   tmp = (tmp << 10) ^ (spill << 50) ^ (spill << 33) ^ (spill << 19) ^ spill;
   return tmp;
}
```

Figure 3: Code for the round constant generating function PSI (Ψ).

In order to use the same circuit for encryption and decryption, it only remains to address the round tweaks. This is done next.

2.9 The Tweak Schedule

We define an *alternating-tweak* schedule for a two-block tweak, where the tweak blocks are independently updated after each use. The update functions are applied during the forward and backward rounds unchanged, i.e., without reversing the schedule as in `QARMAv1`. In this way, any relation between the tweak updates and the round function will have to work with the inverse round function as well, reducing its likelihood.

Let t_i be the round tweak added at round number i. The schedule is the following: $t_1 = T_1$, $t_2 = \varphi^{r-1}(T_0)$, and for each integer $i \geq 1$ we set $t_{2i+1} = \varphi(t_{2i-1})$ and $t_{2i+2} = \varphi^{-1}(t_{2i})$ for some function φ. We obtain:

$$\big[\, T_1,\, \varphi^{r-1}(T_0),\, \varphi(T_1),\, \varphi^{r-2}(T_0),\, \varphi^2(T_1),\, \varphi^{r-3}(T_0),\, \ldots\, ,\, \varphi^{r-1}(T_1),\, T_0 \,\big]$$

where each value in the sequence is used in a successive round from Round 1 to Round $2\,r$. Swapping T_1 and T_0 and applying the same sequence of transformations gives the reverse schedule. Combining this with the inversion of the key schedule, the same circuit can be used for encryption and decryption. We use the following tweak shuffles:

$$\varphi = \tau_f \;:\; \left[\begin{array}{cccc} 1,\,10,14,\,6, & 2,\,9,\,13,\,5, & 0,\,8,\,12,\,4, & 3,\,11,15,\,7 \end{array}\right] \quad \text{for } \ell = 1, \text{ and}$$

$$\varphi = \tau_F \;:\; \left[\begin{array}{cccc} 1,\,10,14,22, & 18,25,29,21, & 0,\,8,\,12,\,4, & 19,27,31,23, \\ 17,26,30,\,6, & 2,\,9,\,13,\,5, & 16,24,28,20, & 3,\,11,15,\,7 \end{array}\right] \quad \text{for } \ell = 2\ .$$

The process we followed to arrive at these choices is described in Section 3.3.

2.10 The Complete Circuit

The circuit, as given in Algorithm 1, has one input bit as well as seven input registers and one output register. The input bit determines whether the circuit performs encryption or decryption. The input registers are the two key blocks, the two center key blocks, the two tweak blocks, and the input text (plaintext PT or ciphertext CT). The output is the output text (ciphertext CT or plaintext PT, respectively). For a key $K = K_1 \| K_0$ and a tweak $T = T_1 \| T_0$, where the K_i and the T_i are blocks, we have:

- For encryption, the input vector is set as follows, with $W_0 = o^2(K_0)$, $W_1 = o^{-2}(K_1)$:

$$\left[\,\mathsf{enc}\,,\, K_0\,,\; K_1\,,\; W_{r+1 \bmod 2}\,,\; W_{r \bmod 2}\,,\; \varphi^{r-1}(T_0)\,,\; T_1\,,\; PT\,\right]\;.$$

- For decryption, the input vector is set to

$$\left[\,\mathsf{dec}\,,\, o^{-1}(K_1)+\alpha\,,\; o(K_0)+\beta\,,\; W_{r \bmod 2}\,,\; W_{r+1 \bmod 2}\,,\; \varphi^{r-1}(T_1)\,,\; T_0\,,\; CT\,\right]\;.$$

The input vector can be set up by a simple wrapper circuit.

2.11 Stretching Shorter Keys

For $\ell = 1$, the only admissible key size is 128 bits.

For $\ell = 2$, we note that the encryption algorithm is always defined with two full 128-bit inputs for the key, i.e. for a key of 256 bits. Only the security margins change, together with the value of the parameter r, as defined later in Section 4. We recommend to randomly generate full 256-bit keys, regardless of the targeted security level, similarly to `BipBip` [BDD+23]. The test vectors given in Appendix I use full 256-bit key values.

This said, we define procedures to *stretch* 128 and 192 bit keys to 256 bits, for interoperability reasons. For a 128-bit key K, set $K_0 = K_1 = K$. For a 192-bit key K, write $K = Y_0 \| Y_1 \| Y_2$ where the Y_i are 64-bit values, and then put $K_0 = Y_0 \| Y_1$ and $K_1 = Y_2 || \big(\mathtt{MAJ}(Y_0, Y_1, Y_2) \ggg 17\big)$, where `MAJ` is the majority function.

2.12 Stretching Single Block Tweaks

`QARMAv2` is defined and analyzed in the case where the tweak is formed by two independent blocks, which we write as $\mathscr{T} = 2$. This said, we also want to understand its security margins when a single block tweak is used, a case we denote with $\mathscr{T} = 1$, that may be interesting for some critical applications. For this purpose we need to define how we "stretch" the b-bit value to a $2\,b$-bit value, and then repeat part of the cryptanalysis.

For $\mathscr{T} = 1$ we set $T_1 = \varphi(T_0)$, where $T_0 = T$. Equivalently, the two tweak blocks just before the center for encryption (Rounds $r-1$ and r) are equal, for both odd and even r.

We tried various approaches to derive the second tweak block from the first, and we chose the one that gave the simplest expression between the two tweak blocks among those that gave the best bounds in related tweak cryptanalysis (see Tables 9c and 9d).

3 Choice of the Building Blocks and Their Properties

3.1 The Orthomorphism

A useful property of orthomorphisms is the following one:

Theorem 1. *Over characteristic 2 algebras, if $o(\cdot)$ is an orthomorphism, then so is $o^2(\cdot)$.*

Proof. First, note that $o^2(\cdot)$ is clearly a bijective linear map. By definition, $p(x) = x + o(x)$ is a bijection, and so is its iterate: $p^2(x) = (x+o(x))+o(x+o(x)) = x+o(x)+o(x)+o^2(x) = = x + o^2(x)$. Hence, $o^2(\cdot)$ is an orthomorphism. □

Therefore, $o^4(\cdot)$, $o^8(\cdot)$, $o^{2^i}(\cdot)$ are orthomorphisms as well. *In particular, none of these maps can be the identity map.* If they operate on a finite algebra, there are only finitely many such maps. This implies also that for $i \neq j$, $o^{2^i}(x) \mapsto o^{2^j}(x)$ is an orthomorphism or the identity, and $x \mapsto o^{2^i}(x) + o^{2^j}(x)$ is a linear bijection or the zero map. Such functions can be used for key derivation to make expected differences between round keys uniform and collision-free, reducing correlations between similar functions. See also Section 4.9.

3.2 The `StateShuffle`

The `MIDORI StateShuffle` [BBI+15] τ (as well as $\overline{\tau}$) satisfies a very important property (that we have not found mentioned elsewhere): *The four cells in each column are mapped by τ to pairwise different columns and rows, and the same holds for the four cells in each row.* Thus, the four elements in each row (or column) of a state after applying τ come from four different columns and from four different rows of the state before τ was applied. Consequently, a following `MixColumns` does not mix any two elements from the same column (or row) of the state before τ was applied. The mapping has order 4.

We observe that the square function of τ,

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

is the conjugate of the transpose of the state by a permutations of the rows.

In [ABI+18], state shuffles with heavier optimal linear trails than τ are listed. We did not to switch to one of these shuffles for a few reasons. First, we rely on the above strong property while studying the interaction of the state and tweak shuffles, for both columns and rows. Second, we prioritize shuffles that achieve faster full diffusion, specifically those that take three rounds (for $\ell = 1$) in conjunction with the chosen diffusion matrix, instead of four. For the limited number of rounds considered, τ, i.e. (2), remains optimal in most cases. Lastly, a slight increase in the weight of optimal linear trails does not necessarily imply an improvement in related-tweak differential characteristics. To serve our needs, we would have to recalculate the counts. Thus, we focus on the tweak shuffles instead.

3.3 The Tweak Shuffles

A TBC requires more rounds than a non-tweakable block cipher based on the same round function to achieve similar security levels. This is because related-tweak attacks must be taken into account. Estimating the number of active cells in linear trails and differential characteristics is vital for selecting the key and tweak schedules. Existing evaluations of `ShiftRows` alternatives [ABI+18] focus on linear trails and single-key, single-tweak differential characteristics, using Matsui's Algorithm 1 [Mat94]. We adapted the latter to include tweaks, but its efficiency degrades significantly due to the much higher number of initial states and possible transitions. As a result, MILP solvers [MWGP11] or SAT solvers like CryptoMiniSAT [SNC09] are much faster.

3.3.1 Single Layer Blocks

The `MANTIS`/`QARMAv1` tweak shuffle was the result of an extended search on a subset of all permutations. The active cell counts on five and a half rounds of the cipher were determined using a MILP model for "several thousand choices for the permutation h" [BJK+16]. Among the shuffles reaching the maximum cell count of 16, one was chosen maximizing the active cells in $\texttt{MANTIS}_5$ first, and then in $\texttt{MANTIS}_6$, namely

$$h \ : \ [\ \ 6,\ 5,\ 14, 15, \quad 0,\ 1,\ 2,\ 3, \quad 7,\ 12, 13,\ 4, \quad 8,\ 9,\ 10, 11 \ \ ] \ , \tag{9}$$

which is the product of two cycles of lengths 2 and 14 and has a period of 14.

QARMAv1 and QARMAv2 have very different tweak schedules. QARMAv1's tweak shuffle is not necessarily optimal also for QARMAv2, and we may want to pick a better one if not.* The search uses a cell-wise MILP model of the cipher. With respect to QARMAv1, QARMAv2's longer tweak translates to a higher number of variables and relations in MILP models. These then take a significantly longer time to solve, and we can only test fewer shuffles in the same time. So, we have to rely more on heuristics. After running several experiments, we observe that following types of shuffles seem to lead to higher active cell counts:

1. Shuffles with fewer and longer cycles;
2. Shuffles leading to schedules without structured, fixed or repeated state patterns; and
3. Shuffles that map aligned groups of cells – i.e. cells that lie either in the same column or in the same row – to aligned groups as much as possible.

With this intuition we immediately found a promising family of shuffles. These are of order 16 and are obtained by permuting all rows cyclically, for instance by sending row i to row $i+1$, with the index considered modulo 4, and then applying a simple rotation to a single row (usually the top one), or three distinct cell swaps involving just two rows, similarly to h. Of these the following stands out after our lengthy MILP runs:

$$h_4 \ : \ \left[\ 13, 14, 15, 12, \quad 0,\ 1,\ 2,\ 3, \quad 4,\ 5,\ 6,\ 7, \quad 8,\ 9,\ 10, 11 \ \right] \ . \tag{10}$$

Table 9a is a comparison of h to this map and to other permutations.

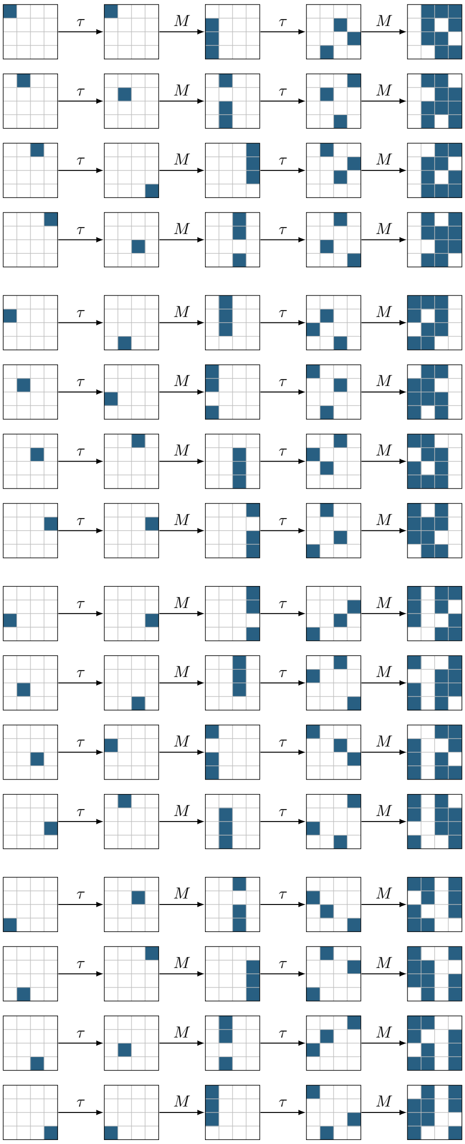
A study of cancellation patterns leads to a better family of shuffles. Because of the alternating tweak schedule, we consider how a cell affects the state after *two* rounds. Let t_n be the round tweak added at round n, such that $t_{n+2} = \varphi(t_n)$ is added at round $n+2$. Suppose the i-th cell of the state is activated by the tweak addition in round n. In order to avoid self-cancellations, the tweak shuffle should not map the t_n's i-th cell to a position $\varphi^{-1}(i)$ that, in the state, is affected by the i-th cell of the state in round $n+2$. In Figure 4a we show how an active single cell evolves through two forward rounds, for all sixteen cells in a layer. We see that after two rounds, seven cells are still unaffected. Ideally, a tweak shuffle φ should map each cell to one of the seven cells it does not affect.

Note that, even if φ is optimal in this sense, the map φ^{-1} used to derive t_{n+3} from t_{n+1} is not guaranteed to be optimal. In fact, most likely it will not be, but we conjecture that φ^{-1} behaves like a randomly chosen permutation. The tweak schedule progresses beyond the reflector unmodified, so φ^{-1} becomes the "optimal" map and φ the "mostly randomly-behaving" one. As we shall see, this strategy seems to be working.

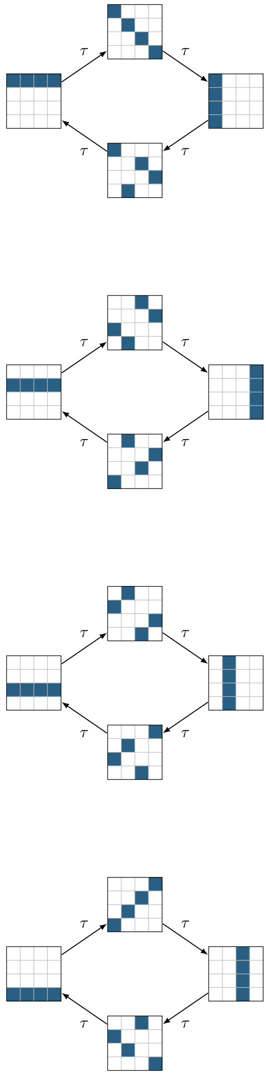
In Figure 4b, we display how τ transforms entire rows and columns. Beside the remark already made in Section 3.3 that τ^2 maps rows to columns (and vice versa) we note also that: *The square of the composite map $M \circ \tau$ lets each cell of a given row act on only nine cells of the state, leaving in particular a single entire column always unaffected. Any two cells in the same row will affect all twelve cells in the three affected columns. The unaffected column is the very one to which the cells of the row are mapped under τ^2.*

This seems to suggest to use τ^2 as the tweak shuffle. However, this does not work well, because τ^2 is an involution, leading to many fixed states and short iterative characteristics. Note that we could permute the cells inside each column of τ^2 independently, leading to $(4!)^4 = 331776$ different shuffles for which the no-self-cancellation property still holds. This is too large a search space. Hence, we proceed as follows:

*Furthermore, under h two tweak cells are only combined with the two corresponding cells of the key, creating a partitioning of the tweak and key bits with a very small group, resulting in low weight algebraic relations between these few bits. Even though we could not find any exploitable such relation, we want to find alternative shuffles with a single cycle or few cycles of similar length.



(a) One active cell through two rounds.



(b) Rows and columns through τ.

Figure 4: Modeling State Transitions for Determining Good Tweak Shuffles.

1. We apply the 24 row permutations to τ^2. The order of the resulting shuffles is at most 8 and only six of them decompose as a product of two cycles of length 8.

2. Among these six shuffles, we select the permutation $\widehat{\tau}$ with the highest active cell counts in related-tweak characteristics for the full-cipher with $r = 5$ and $r = 6$.

3. We look at the order 16 cyclic shuffles obtained by composing $\widehat{\tau}$ with a single cell swap within a column. This produces a set of six shuffles, τ_1 to τ_6 (cf. Table 8).

4. We compute the weights of optimal related-tweak characteristics for the full cipher with $r = 5$ and $r = 6$ for each τ_i and its inverse, $1 \leq i \leq 6$. The shuffle

$$\tau_4 \ : \ [\ \ 2,\ 10, 14,\ 6, \quad 1,\ 9,\ 13,\ 5, \quad 0,\ 8,\ 12,\ 4, \quad 3,\ 11, 15,\ 7\ \ ]$$

maximizes the counts for $r = 6$ and the sum of values for $r = 5$ (cf. Table 9a[†]).

5. Finally, among the order 16 shuffles obtained by applying a single cell swap inside a column to τ_4, we again select one with the best weights in related-tweak differential characteristics, for full-cipher models with $r = 5$, then 6. This shuffle is obtained from τ_4 by swapping the topmost two cells in the first column:

$$\tau_f \ : \ \left[\ 1,\ 10, 14,\ 6, \quad 2,\ 9,\ 13,\ 5, \quad 0,\ 8,\ 12,\ 4, \quad 3,\ 11, 15,\ 7 \ \right] . \tag{11}$$

3.3.2 Two-Layer Blocks

The chosen function is derived from τ^2 as well. We start with τ_f, acting on each layer in parallel, then we swap some cells in the same position between the two layers, provided the order of the resulting shuffle is 32. In our experiments, swapping about a half of the cells leads to better shuffles, but we must swap an odd number of cells to get an order 32 map. We obtain the best results among the sampled shuffles by exchanging the full second and fourth rows, and the cells in position 3 and 19:

$$\tau_F \ : \ \left[\begin{array}{cccc} 1,\ 10, 14, 22, & 18, 25, 29, 21, & 0,\ 8,\ 12,\ 4, & 19, 27, 31, 23, \\ 17, 26, 30,\ 6, & 2,\ 9,\ 13,\ 5, & 16, 24, 28, 20, & 3,\ 11, 15,\ 7 \end{array} \right] . \tag{12}$$

3.3.3 Full Diffusion Properties

Theorem 2. *In* `QARMAv2` *with $\ell = 1$, any input bit nonlinearly affects all bits of the state after three rounds, intended as the first half round, two full rounds, and the diffusion layer of the following round.*

For $\ell = 2$, any input bit nonlinearly affects all bits of the state after four rounds, defined similarly to the $\ell = 1$ case, i.e. the first half round, three full rounds, and the diffusion layer of the following round.

The first claim is inherited from `QARMAv1`. The second claim is proved in Appendix E.

3.3.4 Two-Layer Blocks and `QARMAv1`-128

Remark 3. The two-layer construction of `QARMAv2`-128 is in fact just a variant of `QARMAv1`-128, which has a single layer and 8-bit cells. Upon pairing cell i with cell $i + 16$ in the state and tweak blocks, i.e. a cell in the first layer with the corresponding one in the second layer, we see that all operations work on these 8-bit cells. In particular, τ_F operates like τ_f followed by four bit rotations of nine cells.

`QARMAv2`-128 achieves full diffusion in four rounds instead of three, making it apparently less efficient than `QARMAv1`-128. Note that this fact only impacts how the reach of a distinguisher can be extended, and our cryptanalysis considers it. By considering linear trails and differential characteristics involving 32 4-bit cells, instead of 16 8-bit cells – *with the same linear and differential probabilities* – we obtain better security estimates, more than offsetting the round lost to slower diffusion. With `QARMAv1`-128, improving the security estimates would have required a prohibitively slow bit-wise MILP model.

Our approach is also similar to the one used to construct `AESQ` [KLMR16] and `Pholkos` [BLLS22]. These methods sit between two common approaches. The first one doubles the size of a S-Box by placing two copies of the S-Box side by side and mixing the input and output bits, as in `MIDORI` and `QARMAv1`. The second one uses a generic size-doubling construction on an unmodified cipher [HI19, BCF[+]22]. The strategy used here allows a better analysis, and possibly better security, than the first approach, and it has the potential to perform much better than the second.

[†]Note that sometimes for a given series of full-cipher runs with a fixed φ, the minimum active cell count is *not* a monotonic function of r. The reason for this is explained in Appendix A.2.

3.4 The S-Boxes

As we have seen in the definition of the general purpose cipher, we do not reuse an S-Box from `QARMAv1`. The reason for this is that if we use `QARMAv1`'s σ_1 [Ava17] together with $M_{4,1}$, as we did in an early version of `QARMAv2`, then the unkeyed round function admits a nonlinear invariant that is invariant under translation by a set of 2^{32} constants, as proved by Tim Beyne [Bey23]. While this invariant does not seem to threaten the security of the full cipher, we prefer to avoid such invariants in our design.

Using `QARMAv1`'s σ_0 would have prevented such invariants, but it has less optimal cryptographic properties and gives integral cryptanalysis one extra round of reach. We still use it for the PAC and Memory Authentication versions of the cipher. `QARMAv1`'s σ_2 would have worked as well, but it is much larger and slower. An alternative could have been to keep σ_1 and revert to Class II matrix $M_{4,2}$. However, with $M_{4,2}$ the differential properties of the cipher are significantly worse than with $M_{4,1}$, more so than in `QARMAv1`. So, we need a new S-Box to be used with $M_{4,1}$.

The S-Box ק has the same optimal cryptographic properties as σ_1, namely:

(i) The maximal probability of a differential is 1/4 and there are 15 such differentials;

(ii) The maximal absolute bias of a linear approximation is 1/4 and there are 30 such linear approximations;

(iii) Each of the 15 non-zero component functions has algebraic degree 3;

(iv) Each input bit of the S-Box influences each output bit non-linearly.

The search for ק, like that for σ_1, imposes additional constraints, namely:

(v) For each output bit of the S-Box, and of its inverse, either the SOP (sum-of-products) or the NOT-SOP of its logic negation are sums of at most four monomials;

For each output bit i, let d_i be the minimum of the sums of the degrees of the monomials of the two functions considered in Property (v). Two further conditions are:

(vi) All d_i's of the output bits of the S-Box and of its inverse are bounded by 10; and

(vii) The sum of the eight d_i must be small as possible. The minimum we found was 64. For S-Boxes attaining this minimum, we sieve according to the total weight of the ANFs of the eight output bits of the S-Box and of its inverse, barring constant terms, that we bound by 50.

An additional property is required for ק:

(viii) The S-Box, together with the cipher's linear layer, should not give any chain of invariant affine subspace trails as described in Section 4.7.

We also verify that the composition of S-Box and `MixColumns` has no non-linear invariants, no closed-loop invariants over two rounds, no closed-loop quadratic invariants up to four rounds [WYWP18, WRP20], and no joint symmetric invariants following [Bey18].

Testing random S-Boxes yields very few results after the first few of the above conditions. Hence, we gradually restricted the search to families of potentially good candidates, until we focused on products of four cycles. The reason is that we want to reduce the maximum number of non-linear invariants that the S-Box and linear layer may have in common. Following [Bey18], the number of invariants of the S-Box is related to the number of its cycles, whence it makes sense to minimize the latter. The search is then further limited to cycle structures $(5, 4, 4, 3)$ and $(5, 5, 3, 3)$. The cycle structure of ק is $(5, 4, 4, 3)$. With other four-cycle structures, S-Boxes fulfilling Condition (vii) were found, but they failed Condition (viii). With fewer cycles we only found markedly larger and slower S-Boxes.

3.5 Alternative Constructions

We also experimented with a symmetric tweak schedule, similar to QARMAv1's, where the tweak schedule is reversed in the backward rounds. It used two distinct "optimal" tweak shuffles for T_0 and T_1 that were chosen to minimize repeated mutual cancellations.

For the two-layer version of the cipher, we explored variants with single-row swaps instead of two, and also with the swaps occurring every round or every three rounds.

Across all these variants, the number of active cells was consistently lower for all tested parameter choices, meaning that they would be more advantageous for an attacker.

This seems to point towards a general principle that shuffling only after full diffusion has occurred may result in weaker cryptanalytic properties. Mixing the parallel paths too early would be also sub-optimal. The base, single layer, QARMAv1/QARMAv2 design requires three rounds for full diffusion, and thus allows more options than the AES construction, that requires only two rounds.

4 Security Analysis

In this section, we perform a security analysis of the QARMAv2 design. This leads to the choices for the number of rounds for each targeted security level, as summarized in Table 1a for $\mathscr{T} = 2$ and Table 1b for $\mathscr{T} = 1$. There, ε is a small number, such as 2, to absorb simple optimizations of the attacks, but for the purpose of standardization it may be adjusted to values such as 16 (cf. the NIST requirements for lightweight primitives [NIS] and the corresponding call for algorithms [NIS18]). The rounds are counted as S-Box layers.

Table 1: Security claims and parameter choices.

(a) With two independent tweak blocks ($\mathscr{T} = 2$).

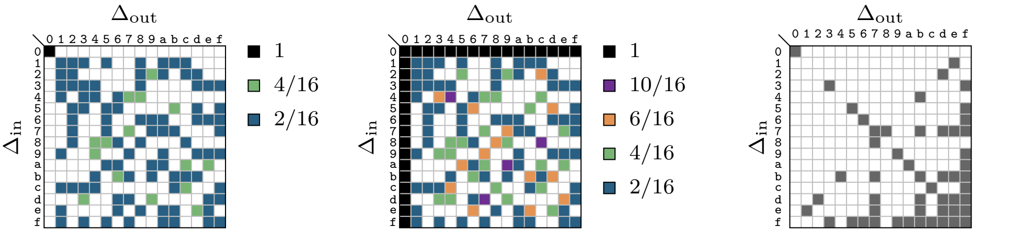
Variant	Block Size	Key Size	Time	Data	Parameter	Rounds
QARMAv2-64-128	64 bits	128 bits	$2^{128-\varepsilon}$	2^{56}	$r = 9$	20
QARMAv2-128-128	128 bits	128 bits	$2^{128-\varepsilon}$	2^{80}	$r = 11$	24
QARMAv2-128-192	128 bits	192 bits	$2^{192-\varepsilon}$	2^{80}	$r = 13$	28
QARMAv2-128-256	128 bits	256 bits	$2^{256-\varepsilon}$	2^{80}	$r = 15$	32

(b) With a single block tweak ($\mathscr{T} = 1$).

Variant	Block Size	Key Size	Time	Data	Parameter	Rounds
QARMAv2-64-128	64 bits	128 bits	$2^{128-\varepsilon}$	2^{56}	$r = 7$	16
QARMAv2-128-128	128 bits	128 bits	$2^{128-\varepsilon}$	2^{80}	$r = 9$	20
QARMAv2-128-192	128 bits	192 bits	$2^{192-\varepsilon}$	2^{80}	$r = 11$	24
QARMAv2-128-256	128 bits	256 bits	$2^{256-\varepsilon}$	2^{80}	$r = 13$	28

We propose one 64-bit block variant and three 128-bit block variants with differing key sizes. Notably, we set data limits of 2^{56} blocks per key for the small block size and 2^{80} blocks per key for the large block size. This is consistent with existing calls for algorithms like the NIST call for lightweight cryptographic algorithms. We believe that these limits are ample for real world applications. The complexities of an attack include any offline precomputations, as without them the attack itself would not be feasible in the first place.

The rest of this section contains the cryptanalysis that supports our choice of parameters. All the cryptanalysis is performed for the version of the cipher with $\mathscr{T} = 2$, unless explicitly noted. Note that any lower bound obtained for $\mathscr{T} = 2$ applies a fortiori to $\mathscr{T} = 1$, as the latter has more restrictions. Table 2 summarizes the results of the various types of cryptanalysis. The parameters for $\mathscr{T} = 1$ are optional.



(a) DDT of the 4-bit S-Box ק. **(b)** BCT of the 4-bit S-Box ק. **(c)** Cell-wise DDT of Class I.

Figure 5: Differential properties of the S-Box ק and of `MixColumns`.

The cryptanalysis is also for the most part performed cell-wise, necessitating only a subset of S-Box properties. While employing a specific S-Box structure, the default assumption is that the S-Box is ק, unless explicitly stated otherwise.

The small version of `QARMAv2` is always analyzed first. If the analysis is purely cell-wise, it can be directly extended to the large version of the cipher, with possible adjustments to the number of rounds for full diffusion or to the active cell counts.

We always assess attacks favorably for the attacker. For instance, we assume that bounds for forward round-only distinguishers hold for symmetric ones as well (except for boomerang distinguishers), and extend any distinguisher by the maximal number of rounds that does not ensure full diffusion.

Finally, we do not claim security in the related-key model, so related keys need to be avoided at the protocol or implementation level. Analyzing related-key attacks on `QARMAv2` may shed light on the necessary level of hardening for protocols or key management.

Table 2: Estimated reach of various types of cryptanalysis.
Values are for $\mathscr{T} = 2$, except numbers in parentheses, which are specific for $\mathscr{T} = 1$.

	`QARMAv2-64`		`QARMAv2-128`		
Attack	Parameter r	Rounds	Parameter r	Rounds	Section
Differential	6 (5)	14 (12)	9 (8)	20 (18)	4.1
Boomerang (Sandwich)	7 (5)	16 (12)	10 (8)	22 (18)	4.2
Linear	5	12	7	16	4.3
Impossible-Differential	3	8	4	10	4.4
Zero-Correlation	3	8	4	10	4.4
Integral (Division Property)	–	5	–	–	4.5
Meet-in-the-Middle	–	10	–	12	4.6
Invariant Subspaces	–	5	–	6	4.7
Algebraic (Quadratic Equations)	–	6	–	7	4.8

4.1 Differential Cryptanalysis

4.1.1 Basic Properties and Cell-Level Bounds

The maximum differential probability (MDP) of `QARMAv2`'s 4-bit S-Box ק is 2^{-2}, and it is reached by 15 differential transitions (18 for σ_0). The complete difference distribution table (DDT) is illustrated in Figure 5a. The boomerang connectivity table (BCT) [CHP+18] in Figure 5b has a maximum probability of 10/16.

We bound the maximum probability of differential characteristics for `QARMAv2` with the help of a MILP model of the cell-wise differential behavior of M. For a detailed description of the modeling approach, we refer to Appendix A.1. The bounds for all variants are

Table 3: Minimum number of active S-Boxes for related-tweak differential characteristics and linear characteristics for `QARMAv2`. A bound of s active S-Boxes implies a maximum probability (or squared correlation) of 2^{-2s} for differential (or linear) characteristics.

(a) With two independent tweak blocks ($\mathscr{T} = 2$).

		Half-Cipher										Full-Cipher				
	r =	3	4	5	6	7	8	9	10	11	12	2	3	4	5	6
ℓ	*Rounds* =	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*6*	*8*	*10*	*12*	*14*
1	Diff.	2	4	8	12	16	22	24	27	32	36	5	12	24	32	41
	Linear	16	23	30	35	38	41	50	57	62	67	5	32	50	64	72
2	Diff.	2	6	11	17	26	34	44	50	*55*	*59*	5	16	32	52	*61*
	Linear	16	25	36	48	58	68	72	80	88	100	24	44	56	80	*96*

(b) With a single block tweak ($\mathscr{T} = 1$).

1	Diff.	5	9	14	19	23	28	31	36	40	45	6	24	32	39	47
2	Diff.	5	12	20	29	41	49	59	67	–	–	6	26	44	67	–

summarized in Table 3. We list both results for the simple iterated round function with an additional initial half round ("half-cipher", where r corresponds to $r+1$ S-Box layers) and for the full reflective construction ("full-cipher", where r corresponds to $2r+2$ S-Box layers). The differential bounds are in a related-tweak setting, while the linear bounds are identical to single-key, single-tweak differential bounds (this is proved in Section 4.3.1).

With the same "good" shuffle, a $2r+2$ rounds half-cipher and a $2r+2$ rounds full-cipher have a similar number of active cells for sufficiently large r (refer to Table 9a). This implies that the reflector construction per se does not significantly change security with respect to linear or differential cryptanalysis. Nevertheless, especially for smaller numbers of rounds, the weights initially ramp up faster in the full-cipher. This may provide an advantage for round-reduced versions for special applications that require security compromises.

4.1.2 Key Recovery

To estimate the amount of rounds an attacker can append to a distinguisher, we analyze the differential distinguisher for `QARMAv2`-64 and $r = 4$ in Figure 11. While this distinguisher has 26 active cells, which is two more than the optimum of 24, it has no active cells in the state which allows us to append more rounds. If we assume that for this truncated differential pattern an optimal characteristic with $p = 2^{-52}$ exists, we are able to sketch a key-recovery attack for $r = 6$ by appending two rounds each at the beginning and end of the distinguisher.

For this attack, we generate two structures of size 2^{53} with 2^{52} pairs each. Then for each structure, we sort the ciphertext based on the 12 inactive bits into buckets. For each pair in each bucket we generate the set of keys compatible with the required difference at the end of the distinguisher. As guessing all involved key-bits would require more than 2^{128} operations, we instead iterate over the $\leq 2^{15}$ differences after one round was appended and then generate the keys that are compatible with these differences. The number of differences is at most 2^{15} because there are five active cells and after the S-Box each active cell can have at most $8 = 2^3$ differences. After we fix this difference one round past the distinguisher, we expect on average one candidate for the 13 active cells in $o(K_1)$ and on average $2^{5.7}$ candidates for the five active cells in $M \cdot o(K_0)$. The expected number of candidates for $o(K_0)$ is higher because we fixed the difference to a known good value. Then, for the rounds added before the distinguisher, we need to guess 35 extra bits of K_0

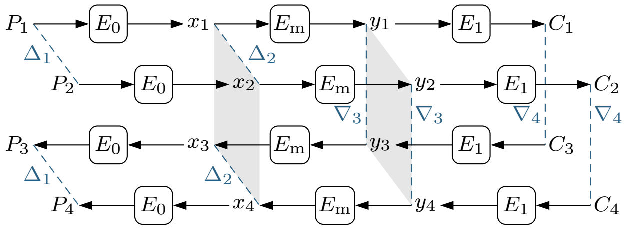


Figure 6: Boomerang/sandwich distinguisher.

and three extra bits of $M \cdot K_1$. By independently verifying the guess in each column, we can keep the complexity below 2^{128}. Finally, after the guesses have been verified, we can increase a counter for each surviving key candidate and then pick the candidate with the highest counter. We estimate that this attack needs 2^{54} encryption queries and 2^{124} time. While the time complexity of this sketched attack could be slightly decreased and also depends on the specifics differences in the distinguisher, we believe that is not possible to add more than two rounds on each end to a differential distinguisher.

For the two-layer version, adding three rounds on each end might be possible because full diffusion is only reached after four rounds.

In conclusion, we expect that `QARMAv2`-64 with $r = 7$ resists differential attacks when data is limited to 2^{56} blocks. For 128-bit blocks with $\ell = 2$, we expect that no differential distinguisher exists for $r = 7$ when data is limited to 2^{80} blocks, and thus that the block cipher with $r = 11$ is secure against differential attacks with data $\leq 2^{80}$ and time $\leq 2^{256}$. The parameters in Table 1 are thus significantly conservative, to take optimizations such as clustering into account [DEKM16, EK18]. In fact, extrapolating the number of active cells in optimal related-tweak differential characteristics with $\mathscr{T} = 2$, we see that for `QARMAv2`-128 with $r \geq 7$ this number most likely exceeds 64, and thus no attack would be possible for $r = 10$ even if the data limit were set to 2^{128} without considering clustering. We posit that an attack with $r = 11$ should then be unfeasible.

4.2 Boomerang Attacks

We bound the probability of characteristics for sandwich distinguishers with the help of a MILP model for $\ell = 1$. The case $\ell = 2$ seems currently unfeasible except for very small values of r. We consider sandwich distinguishers where the cipher is decomposed into three parts $E_1 \circ E_\mathrm{m} \circ E_0$, as illustrated in Figure 6. Since E_m covers many rounds, we assume that the reflector G is part of E_m.

For the MILP model, we roughly follow the approach from [HNE22]: The outer parts E_0, E_1 are modeled with the same cell-wise probabilistic differential model as above, while the inner part E_m models deterministic differential propagation (i.e., truncated propagation with probability 1) in both directions to identify jointly active S-Boxes. The overall probability $p^2\, v\, q^2$ of the sandwich distinguisher can then be derived based on the differential probability of the S-Boxes in the outer rounds (bounded by the MDP of 4/16) for p, q and the boomerang probability of the jointly active S-Boxes in the inner rounds (bounded by the maximum nontrivial entries in the BCT of 10/16) for v.

For simplicity, instead of weights $2 \cdot 2 = 4$ in the outer part and 10/16 in the inner part, we scale the weights to 6 in the outer part and 1 in the inner part, so the final bounds in Table 4 need to be scaled by a factor of about 2/3 to get the boomerang probability. Since we are analysing a reflector cipher, we not only have to evaluate different combinations for the round number of E_0, E_m, E_1, but also configurations that differ on where the reflector is located. We also optimize the position of the split between E_0, E_m, E_1 to be in the middle of a round, since this yields higher probabilities. Our results are given in Table 4.

Table 4: Bounds for sandwich characteristics for `QARMAv2`-64 with r_F forward and r_B backward rounds. An entry x corresponds to a boomerang probability of about $2^{-(2/3)x}$. A likelihood of 2^{-56}, resp. 2^{-80} is reached when the entry is equal to 84, resp. 120 or higher. Numbers in italics are linearly extrapolated. Note that for $r_\mathrm{F} = r_\mathrm{B}$ we can assume $r = r_\mathrm{F} - 1$ and extend the reach of a symmetric distinguisher by four, resp. six rounds for $\ell = 1$, resp. 2, and progressively less with the increase of the difference between r_F and r_B.

(a) With two independent tweak blocks.

$r_\mathrm{F} \backslash r_\mathrm{B}$	1	2	3	4	5	6	7	8	9	10
1		6	6	6	6	6	15	21	32	44
2	6	6	6	6	7	15	21	32	44	60
3	6	6	6	8	18	24	35	47	62	76
4	6	9	15	28	30	38	48	63	76	*92*
5	10	12	18	30	46	58	73	76	*90*	*107*
6	12	15	26	38	51	82	85	*94*	*107*	*123*
7	15	21	35	48	73	97	*105*	*115*	*122*	
8	21	32	49	63	81	*104*	*116*	*126*		
9	32	46	62	75	*93*	*114*	*126*			
10	46	60	78	*93*	*109*	*120*				

(b) With a single block tweak.

$r_\mathrm{F} \backslash r_\mathrm{B}$	1	2	3	4	5	6	7	8	9	10
1		6	6	10	15	24	36	50	64	80
2	6	12	18	22	27	35	48	59	75	*89*
3	6	21	33	42	54	68	85	100	*115*	
4	10	22	45	60	76	90	106	*121*		
5	15	36	57	76	92	108	*126*			
6	23	35	65	92	108	*126*				
7	36	60	88	108	*126*					
8	47	59	89	*110*						
9	67	91	*107*							
10	79	*101*								

In Figures 13 to 15 in the Appendix, we provide selected characteristics. Each entry in Table 4 is the minimum over all possible boomerang configurations for the given round configuration. Optimal outcomes often arise from unbalanced round setups, where the forward or backward parts significantly differ in length: However, they are unlikely to effectively target the full cipher. These bounds rely on cell-wise differential bounds for individual characteristics for the outer boomerang rounds; exploiting a potential clustering effect for specific bitwise differences may slightly increase the resulting probabilities. Conversely, the bound for the inner part E_m is likely too optimistic, and we expect the actual probabilities for conforming characteristics to be lower. Without a separate switching layer, deriving bounds instead as $p^2\, q^2$, we anticipate boomerang attacks to be less effective than differential ones due to the roughly linear growth of differential bounds.

For $\ell = 2$, the bounds must be extrapolated. From Table 3, we can expect that differential characteristics for $\ell = 2$ have roughly 50% more active S-Boxes than those for $\ell = 1$ for the same number of rounds. Extrapolating the growth of the values in Table 4 to reach the value 120 (corresponding to a boomerang probability of $\approx 2^{-80}$), we expect that usable boomerang distinguishers will exist for values of r higher by 2 than the values for $\ell = 1$. Considering the additional round for full diffusion with respect to $\ell = 1$, we bound r for $\ell = 2$ by adding 3 to the bounds for $\ell = 1$.

4.3 Linear Cryptanalysis

4.3.1 Basic Properties and Cell-Level Bounds

The maximum squared correlation of the 4-bit S-Box ק of `QARMAv2` is 2^{-2}, which is reached by 30 linear approximations (32 for σ_0).

The chosen `MixColumns` matrix $M = \mathrm{circ}(0, \rho, \rho^2, \rho^3)$ is involutory and symmetric when viewed as a 16×16 matrix of bits. Since we need to analyze the inverse of the bitwise transpose of the matrix which equals the original matrix, we can reuse the existing model.

Furthermore, since the tweak schedule is completely known by the attacker, it does not contribute any linear bias, and we do not need to model it. Thus, the linear model is equivalent to the differential single-key, single-tweak model. In particular, the linear branch number is 4 and the cell-wise linear behavior is the same as in Figure 5c. The resulting bounds are summarized in Table 3.

For $\ell = 1$, at least 32 active S-Boxes, corresponding to a squared correlation of at most 2^{-64}, are reached after seven rounds of the half-cipher (parameter $r = 6$) or eight rounds of the full-cipher ($r = 3$). For $\ell = 2$, at least 64 active S-Boxes are reached after nine rounds of the half-cipher ($r = 8$) or twelve rounds of the full-cipher ($r = 5$).

4.3.2 Key Recovery

Efficient key recovery for linear distinguishers can take advantage of the FFT technique [CSQ07]. This technique was recently improved further to define an optimized framework [BCFG+21, FGNP20]. Taking the number of rounds for full diffusion (3 rounds for $\ell = 1$, 4 rounds for $\ell = 2$) as a bound for key-recovery rounds in the beginning and end of the cipher, we expect a solid security margin for linear cryptanalysis.

4.4 Impossible-Differential and Zero-Correlation Cryptanalysis

The resistance of QARMAv2 against impossible-differential and zero-correlation cryptanalysis is evaluated using an automated model inspired by the framework in [HSE23].

We split our cipher in two halves: the upper and the lower half. Then, we model propagation with probability 1 in forward/backward direction for the upper/lower half, respectively. As we want to find contradictions, where the two halves meet, we model each cell of the block cipher to be in one of four states: inactive, active with a single difference/mask, active with nonzero difference/mask, and active with unknown (possibly zero) difference. This representation allows us to find contradictions where a cell that is inactive in one half is active with a single or any nonzero difference in the other half.

When running this model for $\ell = 1$, we find that the longest impossible differential distinguisher spans nine rounds as depicted in Figure 12, namely: one half round, four full forward rounds, the reflector, three full backwards rounds and one backward half round. This implies that for $r = 4$ no impossible differential distinguisher exists.

When considering $\ell = 2$, four rounds provide full diffusion; so with $r = 5$, no impossible differential distinguisher exists. This is because we can skip the initial half round and one of the full rounds by leaving the plaintext and one of the tweaks inactive. Due to the full diffusion, when the two halves meet, no miss-in-the-middle is possible.

When analysing the zero-correlation setting with $\ell = 1$ a fixed tweak, the longest distinguisher we can find covers six rounds, i.e. QARMAv2 with $r = 2$.

In the case of a variable tweak with $\ell = 1$, we need to consider that a contradiction might also arise in the tweak schedule. However, for $r = 4$, no zero-correlation distinguisher exists. This is because after three applications of the state shuffle and MixColumns, all linear masks must be active with unknown masks. Then, with another round this implies the same unknown masks in the tweak. As our cipher is symmetric this holds for T_0 and T_1. For $\ell = 2$, the same argument applies for $r = 5$ due to the slower diffusion.

Considering the number of rounds for full diffusion (three rounds for $\ell = 1$, four rounds for $\ell = 2$) as a bound for key-recovery rounds in the beginning and end of the cipher, we expect a solid security margin for impossible-differential and zero-correlation cryptanalysis.

4.5 Integral Cryptanalysis and the Division Property

Integral cryptanalysis operates by identifying properties of a collection of ciphertexts corresponding to predetermined *sets* of plaintexts with a specific structure. It can be viewed as a generalization of differential cryptanalysis, which uses sets of cardinality two.

The attacker typically assembles a collection of N chosen plaintexts and generates the corresponding ciphertexts using the target block cipher. By observing the XOR of the ciphertexts in specific word positions, if the result is $\mathbf{0}$, it indicates the presence of an integral characteristic within the block cipher when subjected to the N chosen plaintexts.

Essentially, integral cryptanalysis exploits low algebraic degrees of a cryptographic primitive, and countering it starts with the proper choice of S-Box, for instance by making sure that all its outputs have sufficiently large degree (see Section 3.4).

The *Division property* is a generalization of the integral property that was proposed by Todo [Tod15]. This tool allows for a more systematic way of ascertaining the existence of integral characteristics by tracing their propagation across the diverse operations within a cipher. While the original division property was word-based, Todo and Morii [TM16] introduced the bit-based variant, to enable a more precise analysis.

Definition 1. *(Bit-based Division Property [TM16])* A multi-set $X \subseteq \mathbb{F}_2^n$ is said to have the division property $\mathbb{K}$ for some set of n-dimensional vectors $\mathbb{K}$ if for all $\mathbf{u} \in \mathbb{F}_2^n$, it fulfills

$$\bigoplus_{\mathbf{x}\in X} \mathbf{x}^{\mathbf{u}} = \begin{cases} \text{unknown, if there is } \mathbf{k} \in \mathbb{K} \text{ s.t. } \mathbf{u} \succeq \mathbf{k} \\ 0, \text{ otherwise} \end{cases}$$

Definition 2. (Balanced Position) Let $Y \subseteq \mathbb{F}_2^n$ be a multi-set of vectors. A coordinate position $0 \leq i < n$ is called balanced position if $\bigoplus_{\mathbf{y}\in Y} y_i = 0$.

The bit-based division property is an effective technique for determining whether a particular monomial is present (the *unknown* case) or absent (the *zero* case) in the polynomial representation of the product of output bits. Once we have successfully identified a set of balanced positions that correspond to a specific input division property $\mathbf{k}$, we can use this property to distinguish a cipher E from a randomly chosen permutation. To accomplish this, we first construct a set X of plaintexts that form an affine subspace, aligning with the input division property $\mathbf{k}$. For each vector $\mathbf{x} = (x_0, x_1, \cdots, x_{n-1})$ within the set X, we set the i-th coordinate to a fixed constant c_i from the binary set $\{0, 1\}$ if the i-th coordinate of $\mathbf{k}$ is 0. If the i-th coordinate of $\mathbf{k}$ is 1, we allow x_i to take on any value within the binary set $\{0, 1\}$. The size of X is $2^{wt(\mathbf{k})}$, where $wt(\mathbf{k})$ is the Hamming weight of $\mathbf{k}$.

While originally a direct programming approach was used to find bit-based division properties for 32-bit block ciphers [Tod17], this is impractical for larger block sizes. Instead, the propagation rules can be encoded for solving with automatic tools, as proposed by Xiang et al. using MILP [XZBL16], and Sun et al. using SAT/SMT [SWW17].

We apply the SAT/SMT approach to `QARMAv2`-64 and convert the propagation rules into Conjunctive Normal Form (CNF) to be solved using CryptoMiniSAT [SNC09]. The longest integral characteristic found covers five forward rounds of the cipher. We did not find any integral distinguishers over six forward rounds. If we use the S-Box σ_0 we find integral characteristics for up to six rounds and none on seven rounds.

4.6 Slide, Meet-in-the-Middle and other Structural Attacks

Slide attacks [BW99] and variants [Kar07] exploit similarity between sequences of rounds. The main idea behind these attacks is that once sequences of rounds induce the same permutation, then one can attack the rounds around the similar permutation by using slid pairs, i.e., pairs of values that have the same input to the same permutation.

For `QARMAv2` the issue is mitigated by several components. The cipher uses four different types of rounds beside the reflector. The round constants are different at each round, and since they are not light nor dense, one cannot find complementation properties or slide properties that hold with significant probability. Finally, even if one succeeds to find a combination of (key, tweak) and round constants such that some forward rounds are the same (in a related-tweak settings), then the cipher's design implies that this similarity should hold through the reflector and the inverse rounds. However, both reflector and inverse rounds use different keys and the reflector is untweaked.

The best MitM attacks on QARMAv1 [ZD16, LZG+20] can possibly be adapted to QARMAv2 and we do not expect them to be more successful. The authors of [ZD16] do not include the time to prepare the data tables in their time complexity. Therefore their 10-round key recovery attack on QARMAv1-64 has data complexity of 2^{53} chosen plaintexts and time complexity of 2^{116}. The MitM attack in [LZG+20] has time, data, and memory respectively $2^{156.06}$, 2^{88} CP, and 2^{154}. Since there is a large gap between the number of attacked rounds and the parameters given in Table 1, we posit that similar attacks are not a threat to the security of QARMAv2.

4.7 Invariant Subspace Cryptanalysis and Non-Linear Invariants

To determine whether QARMAv2-64 amd QARMAv2-128 have invariant subspaces, we can repeat the arguments in [Ava17] or in [LMR15] and conclude that the forward functions of QARMAv2-64, resp. QARMAv2-128, do not have non-trivial invariant subspaces as soon as we consider three, resp. four, full rounds. The problem with this approach is that it does not properly account for the tweak parameter.

Besides the steps taken in the S-Box choice (Property (viii) in Section 3.4) to avoid symmetric non-linear invariants, we can go further and consider arbitrary invariants. Since invariants are support functions of sets of points, we consider their enveloping subspaces.

We start by giving a slight generalization of subspace trail cryptanalysis [GRR16].

Subspace trail cryptanalysis keeps track of subspaces of the state space – in our case $V = (\mathbb{F}_2^4)^d = \prod_{j=0}^{d-1} V_j$ where all $V_j \cong \mathbb{F}_2^4$ and $d = 16\,\ell$ – that remain invariant up to translation under the action of the *unkeyed* round function.

Definition 3. Let $F : V \to V$ be a *round function*, and r a natural number. A sequence of $r+1$ linear subspaces $Z^{(0)}, Z^{(1)}, \ldots, Z^{(r)} \subseteq V$ is called a r-round *subspace coset trail* if for each $i = 1, 2, \ldots r$ there is at least one pair of elements $(a_i, b_i) \in V^2$ such that $F(Z^{(i-1)} + a_i) \subseteq Z^{(i)} + b_i$. The points a_i, b_i are called *input* and *output offsets*.

In order to determine conditions on the *round tweakeys*, we must consider each subspace together with an offset. We call such a (subspace, offset) pair a *coset* for brevity. If a round tweakey is contained in $Z^{(i)} + b_i + a_{i+1}$, then it can be added to the state and the property for a given state value to belong to $Z^{(i)} + b_i$ propagates one more round. Conversely, by observing the input and output values, it can be determined whether the round tweakeys belong to a set of *weak tweakeys* – the sets $Z^{(i)} + b_i + a_{i+1}$, with (hopefully) only a small chance of false positives. Thus, we want to study how subspaces and their cosets propagate through the cipher. Trivial trails where all $Z^{(i)} = \{0\}$ or that end with $Z^{(i)} = V$ are not useful. So, we assume that trails do not start with the zero space and do not end with V.

Our goal is now to prove that such trails cannot extend to distinguishers over more than a few rounds.

Put $Z^{(i)} = \bigotimes_{j=0}^{15} Z_j^{(i)}$ with $Z_j^{(i)} \subseteq V_j$ for all j. If one $Z_j^{(i)} \cong \mathbb{F}_2^4$, then $Z^{(i')} = V$ for some $i' \leq 3 + i$ (for $\ell = 1$) or $i' \leq 4 + i$ (for $\ell = 2$). So we can assume that all $Z_j^{(i)} \subsetneq V_j$.

The dimension 0 and 1 cosets are individual values and pairs of values, respectively. Tracking trails of only these coset types is analyzed more generally, i.e. with likelihoods not necessarily equal to 1, by linear and differential cryptanalysis. Thus, we focus on trails with at least one cell having a coset of dimension at least 2. Using a sage script, we can list the images of the cosets in $\mathbb{F}_2^4$ under ק. No dimension 3 coset maps to another dimension 3 coset, and only following dimension 2 cosets map to other dimension 2 cosets:

$$\begin{cases} \mathtt{2} + \langle \mathtt{4}, \mathtt{9} \rangle \xmapsto{\sigma} \mathtt{9} + \langle \mathtt{3}, \mathtt{4} \rangle \\ \mathtt{0} + \langle \mathtt{4}, \mathtt{8} \rangle \xmapsto{\sigma} \mathtt{0} + \langle \mathtt{4}, \mathtt{8} \rangle \\ \mathtt{5} + \langle \mathtt{2}, \mathtt{8} \rangle \xmapsto{\sigma} \mathtt{3} + \langle \mathtt{5}, \mathtt{9} \rangle \\ \mathtt{1} + \langle \mathtt{5}, \mathtt{B} \rangle \xmapsto{\sigma} \mathtt{1} + \langle \mathtt{6}, \mathtt{B} \rangle \\ \mathtt{3} + \langle \mathtt{7}, \mathtt{A} \rangle \xmapsto{\sigma} \mathtt{2} + \langle \mathtt{7}, \mathtt{9} \rangle \end{cases}$$

The fact that a coset $Z^{(i)}$ is mapped by the round function to a coset $Z^{(i+1)}$, i.e. $Z^{(i+1)} = (S \circ M \circ \tau)\big(Z^{(i)}\big)$, can be expressed via a set of relations on the cells such as

$$Z_0^{(i+1)} = \sigma\Big(\rho\big(Z_{10}^{(i)}\big) + \rho^2\big(Z_5^{(i)}\big) + \rho^3\big(Z_{15}^{(i)}\big)\Big) \ . \tag{13}$$

Let us take a closer look at this relation. As we are chaining states that contain dimension two cosets, $Z_0^{(i+1)}$ and at least one of the $Z_j^{(i)}$ on the r.h.s. have cardinality four. If, say, $\#Z_{10}^{(i)} = 4$, we have $Z_0^{(i+1)} = \sigma\big(\rho\big(Z_{10}^{(i)}\big) + \delta\big)$ where δ is the sum of an element of $\rho^2\big(Z_5^{(i)}\big)$ and one of $\rho^3\big(Z_{15}^{(i)}\big)$. In other words, a cell $Z_j^{(i)}$ contributes to $Z^{(i+1)}$ through σ only after a rotation by one to three bits. The corresponding mapping of subspaces is:

$$\begin{cases} \langle \mathtt{2}, \mathtt{C} \rangle, \langle \mathtt{1}, \mathtt{6} \rangle, \langle \mathtt{3}, \mathtt{8} \rangle \xmapsto{\rho^{\{1,2,3\}}} \langle \mathtt{4}, \mathtt{9} \rangle \xmapsto{\sigma} \langle \mathtt{3}, \mathtt{4} \rangle \\ \langle \mathtt{2}, \mathtt{4} \rangle, \langle \mathtt{1}, \mathtt{2} \rangle, \langle \mathtt{1}, \mathtt{8} \rangle \xmapsto{\rho^{\{1,2,3\}}} \langle \mathtt{4}, \mathtt{8} \rangle \xmapsto{\sigma} \langle \mathtt{4}, \mathtt{8} \rangle \\ \langle \mathtt{1}, \mathtt{4} \rangle, \langle \mathtt{2}, \mathtt{8} \rangle, \langle \mathtt{1}, \mathtt{4} \rangle \xmapsto{\rho^{\{1,2,3\}}} \langle \mathtt{2}, \mathtt{8} \rangle \xmapsto{\sigma} \langle \mathtt{5}, \mathtt{9} \rangle \\ \langle \mathtt{7}, \mathtt{A} \rangle, \langle \mathtt{5}, \mathtt{B} \rangle, \langle \mathtt{7}, \mathtt{A} \rangle \xmapsto{\rho^{\{1,2,3\}}} \langle \mathtt{5}, \mathtt{B} \rangle \xmapsto{\sigma} \langle \mathtt{6}, \mathtt{B} \rangle \\ \langle \mathtt{5}, \mathtt{B} \rangle, \langle \mathtt{7}, \mathtt{A} \rangle, \langle \mathtt{5}, \mathtt{B} \rangle \xmapsto{\rho^{\{1,2,3\}}} \langle \mathtt{7}, \mathtt{A} \rangle \xmapsto{\sigma} \langle \mathtt{7}, \mathtt{9} \rangle \end{cases} \tag{14}$$

This means, for instance, that if a cell contains a coset of $\langle \mathtt{2}, \mathtt{C} \rangle$, $\langle \mathtt{1}, \mathtt{6} \rangle$, or $\langle \mathtt{3}, \mathtt{8} \rangle$ in a given round of a coset trail, in the following round it only contributes to cells containing a cosets of $\langle \mathtt{3}, \mathtt{4} \rangle$. In fact, this coset must be $\mathtt{9} + \langle \mathtt{3}, \mathtt{4} \rangle$ before the addition of any round tweakey. This also explains why the fact that $\sigma(\mathtt{0} + \langle \mathtt{4}, \mathtt{8} \rangle) = \mathtt{0} + \langle \mathtt{4}, \mathtt{8} \rangle$ is not a problem: this set is only used under rotation, and the underlying vector subspace is not invariant under rotations by one, two, or three bits. It is easy to see that the relations in Equation (14) cannot be chained. This means that, at the following round, we have cosets of dimension at least 3, and since these do not map to other dimension 3 cosets, in the third round there is at least one cell with its full dimension 4 space in the subspace trail. Then, after at most three (for $\ell = 1$) or four (for $\ell = 2$) rounds, the trail ends in the whole space V. Hence, no useful trail can extend for more than five, resp. six rounds.

4.8 Algebraic Attacks

We consider the applicability of algebraic attacks [CP02] on `QARMAv2`-b.

As in [Ava17, Section 4.2], `QARMAv2`-b has sufficiently many rounds to reach maximum algebraic degree $b - 1$. In fact, the exact same argument holds, and `QARMAv2`-64, resp. `QARMAv2`-128 reach the upper bound $b - 1$ after seven, resp. eight rounds. `QARMAv2`-64 and `QARMAv2`-128 have at least 16 and 20 rounds in their most aggressive general purpose versions, hence they should have sufficiently many rounds to reach maximum degree.

Let us now count how many quadratic equations and variables are necessary to describe the two variants of `QARMAv2`. Following [CP02], it is straightforward to verify that both

σ_0 and ק are described by $e = 21$ quadratic equations in the eight input and output variables over $\mathbb{F}_2$. Hence, the entire system for a fixed-key `QARMAv2`-b permutation consists of $\frac{b}{4} \cdot (2r+2) \cdot e$ quadratic equations in $\frac{b}{4} \cdot (2r+2) \cdot v$ variables. For `QARMAv2`-64 with $r = 9$ this translates to 6720 equations in 2560 variables, and for `QARMAv2`-128 with $r = 11$ to 16128 equations in 6144 variables. For comparison, a fixed-key `AES`-128 (resp. `AES`-256) permutation consists of 6400 (resp. 8960) equations in 2560 (resp. 3584) variables.

The same conclusions as in [Ava17] should stand, i.e. that `QARMAv2`-64 and `QARMAv2`-128 should offer good resistance against algebraic cryptanalysis if also the `AES` does.

4.9 Security Implications of the Reflector

The arguments in [Ava17] hold in a stronger form. For each addition of K_0 (resp. K_1) on the forward path, $o(K_0)$ (resp. $o^{-1}(K_1)$) is added in the backward path. Hence, any reflection-like relation encompassing the reflector G and any (odd) number of rounds on each of the two sides would be disrupted (i.e. its likelihood reduced) in a similar way as described in [Ava17]. This means that for each $i \in \{0, 1\}$, the map $K_i \mapsto K_i + o^{1-2i}(K_i)$ is bijective (we can ignore the constant α), and thus it does not give rise to potentially equivalent or weak classes of keys. The fact that we add $o^2(K_0)$ and $o^{-2}(K_1)$ at the center allows to apply the same argument with an odd number of rounds at a single side of the reflector. This should reduce the effectiveness of reflection attacks [Kar08, SBY+15].

We also have observed that key recovery is more difficult with the application of orthomorphisms to the keys than without.

4.10 Comparison to `QARMAv1`

`QARMAv2`'s parameter r seems, superficially, larger than `QARMAv1`'s at the same security level. When we will later compare the latencies directly, indeed we lose a bit to `QARMAv1`, but we posit that this seems worthy since we get a longer tweak size in exchange. Furthermore, we have more *robust* security margins in the sense that the 128, 192, and 256 bit levels are defined as time (T) $2^{128-\varepsilon}$, $2^{192-\varepsilon}$, and $2^{256-\varepsilon}$, respectively, to mount a successful attack on the cipher, with the data (D) limits for `QARMAv2` having been chosen conservatively.

With the same data limits, the 128-bit security level for `QARMAv1`-64 corresponds to time $T = 2^{72-\varepsilon}$, and the security levels of 256 bits for `QARMAv1`-128 translate to time $T = 2^{176-\varepsilon}$. These values are actually lower bounds. They are established by noting that `QARMAv1`, while formally a three-round Even-Mansour construction [EM91], is quite close to being single-round due to the simplicity of the reflector. Nevertheless, it is crucial not to dismiss the reflector completely: Taking into account the likelihood of guessing a difference at the sides of the reflector, the complexity may be increased by $2^{b/2}$, hence $T = 2^{104-\varepsilon}$ and $T = 2^{208-\varepsilon}$. In the `QARMAv1` paper, it is shown that an attack to `QARMAv1`-b exist with $T = 2^{7b/4}$, $D = 2^{3b/4-1}$ and memory $M = D$, resulting in $T = 2^{112}$ with $D = 2^{47}$ for `QARMAv1`-64 and $T = 2^{224}$ with $D = 2^{95}$ for `QARMAv1`-128.

In Appendix G we report the published cryptanalysis of `QARMAv1`. No attack so far has broken the security claims for full-round versions of the cipher. We expect that `QARMAv2`, with its improved structure, will be similarly resistant to cryptanalysis, despite the stronger security claims.

5 Version for Pointer Authentication and Memory Integrity

`QARMAv2` can be used as a short-message MAC by simply truncating its output: If $m < b$ bits are removed, the construction is secure up to about $2^{(b+m)/2}$ queries [BI99].

For the PAC feature and the computation of memory integrity tags (following [JLK+23]), we instantiate `QARMAv2`-64 with either ק (see Equation (4)) or `QARMAv1`'s S-Box σ_0 (see

Equation (5)). If the tweak has a length of only one block, or, in the PAC instruction set terminology, only one "salt" is used, then we put $T_1 = \varphi(T_0)$, where $T_0 = T$. The value of the parameter r is 6 for memory integrity and for pointer authentication. σ_0 is to be used in case the latency with ק is too high.

If $r = 6$ cannot be reached for pointer authentication without significant performance penalties, for instance on small in-order cores, then it is admissible to use $r = 5$ or $r = 4$.

5.1 Usage

We now describe how to use QARMAv2-64 for memory integrity. Let us partition a cache line M in 64-bit blocks $M = M_0|M_1|\cdots|M_{r-1}$. We say that a function provides *temporal uniqueness* – or *freshness* – when repeated writes of the function computed on the same plaintext in the same memory location result in different outputs. Freshness is usually achieved by including a counter as an input to the function. Now, depending on whether the memory encryption algorithm offers freshness or not, we can use one of two integrity algorithms in Figures 7 and 8. They are "tPMACs", i.e. tweakable Parallelizable MACs. The first one is for the case where the system does not provide freshness, and the second one is to be used when it does. α_i is the physical address of the block that is being encrypted as a contribution to the memory region's tag, and ν is a nonce or counter.

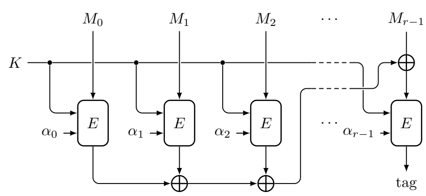


Figure 7: PMAC computed with a TBC when freshness is not provided.

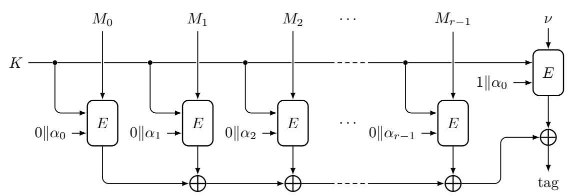


Figure 8: PMAC computed with a TBC when freshness is provided.

The output of the QARMAv2 function needs to be truncated to be used as a PAC. To generate a z-bit tag with $1 \le z \le 32$, first reorder the output bits as follows:

$$[0, 8, 16, 24, 32, 40, 48, 56]\,, \quad 4 + [0, 8, \ldots, 56]\,, \quad 1 + [0, 8, \ldots, 56]\,, \quad 5 + [0, 8, \ldots, 56]\,.$$

Then, pick the first z bits in this sequence.

5.2 Security

Note that QARMAv2-64's output for both PAC and memory integrity is truncated to 32 or fewer bits. The purpose of these features is to deter attacks, not to prevent them with absolute certainty. For instance, a 32-bit memory integrity tag should not be guessable with likelihood higher than 2^{-32}. This applies to the two most common deployments:

1. Encryption and integrity do not provide temporal uniqueness. This prevents memory corruption but does not address replay attacks, forgery, or semantic security violations. An example is Intel's TDX [Int21] (which uses 28-bit integrity tags).

2. Encryption and integrity provide temporal uniqueness by including counters in the inputs. The counters must be themselves protected, for example by an integrity tree as in Intel's SGX [Gue16]. This offers semantic security, and prevents replay attacks.

For a full-cipher with $r = 6$, we can take a differential characteristic for $r = 4$, which has at least 24 active cells, and we (optimistically) extend it by two rounds on each side to cover the whole full-cipher with $r = 6$. However, for an output of 32 bits or less, any distinguishable characteristic must have likelihood higher than 2^{-32}.

For a full-cipher with $r = 5$ (resp. $r = 4$), we need to consider the active cells with $r = 3$ (resp. $r = 2$), which is just just 12 (resp. 5), so differential cryptanalysis cannot be mounted if the tag is truncated to 24 bits (resp. 10 bits) or less. Therefore, any cryptographic attack is here made more difficult only by the fact that it would likely be a known plaintext attack, or that the control that an adversary can have on the input is quite limited. The PAC feature is thus only effective if in these cases the pointer tag is truncated to at most 24, resp. 10 bits. This prevents differential cryptanalysis, and the adversary can only guess the tag bits or try to recover the key by brute force.

Considering linear attacks, for $r = 4$, we already have 50 active S-boxes, hence a 2^{100} data complexity which is even above the data limits for the normal cipher.

6 Hardware Implementation and Evaluation

Our target construction is a low-latency block cipher, hence we evaluate the metrics of a fully unrolled circuit from the input to output ports. We compare our design with the following ciphers: `AES-{128|192|256}`, `MANTIS-{7|8}`, `QARMAv1-{64|128}`, `PRINCE`, `Orthros`, `SPEEDY-{6|7}`, `MIDORI-{64|128}`, `PRESENT-{80|128}`, `SKINNY-{64|128}`, and `BipBip`. We also consider the `ASCON-p`^{12} permutation used in a single key Even-Mansour mode. We skip `PRINCEv2` as latency and area are almost identical to the original `PRINCE`.

We included also `MANTIS`, a 64-bit only precursor to `QARMAv1`, supporting single-block tweaks. The {0,1}-diffusion matrix and the simpler reflector are significant weaknesses of the design. It has been broken for $r \leq 6$ [EK18], but the attacks do not extend to `QARMAv1`.

Finally, we consider `BipBip` [BDD+23], a 24-bit block cipher with a 256-bit key and 40-bit tweak support. It is designed to encrypt the least significant 24 bits of a 64-bit pointer while using the top 40 bits as the tweak, to safeguard against pointer forgery.

In order to guarantee a fair comparison to `SKINNY` [BJK+16], we use the more recent parameters that are used in Romulus for the NIST Lightweight Cryptography Competition [IKMP20, IKMP19]. We observe that in related-tweakey characteristics for `SKINNY-128-384`, a probability not exceeding 2^{-128} is reached first with 26 rounds. If we add then 10 rounds because of the six-round full-diffusion we reach 36 rounds and this means that the 40 rounds used in Romulus represent a margin of 10%. Applying the same reasoning to `SKINNY-64-192`, we see that probability 2^{-64} is achieved after 18 rounds. Adding 10 rounds and a margin of 10% we reach 31 rounds, which we round up to 32, since this is the minimum suggested in the `SKINNY` paper anyway.

6.1 Methodology

We follow the evaluation framework outlined in [BIL+21]. The relative area and latency advantages between different primitives may vary considerably with the manufacturing processes. As a results, a company's or standardization body's business decision of "betting" on a primitive over another cannot be based on a single data point. Hence, we compare our choice of ciphers at three different processes:

1. At 90nm lithography with the STM cell library `CORE90GPSVT.CMOS090LP`;
2. At 15nm lithography with the `Nangate 15nm Open Cell Library` [MMR+15]; and
3. At the TSMC 5nm lithography with the `tsmc_sch280pp57_cln05fb41001` library, courtesy of the Arm implementation team.

For a fair evaluation we adhered to the following design flow for all the ciphers:

1. The ciphers have been implemented in VHDL, and a functional simulation was done using the `Modelsim` software. Correctness has been verified against test vectors.

2. For the 90nm and 15nm processes we synthesize the circuits with *Synopsys Design Compiler*. For the 5nm process, the VHDL code is first converted to Verilog using GHDL with the yosys plugin (available at https://github.com/ghdl/ghdl-yosys-plugin), and then compiled using *Cadence Genus*.

3. Optimizing for area is a computationally heavy stage that outputs a circuit with, usually, near minimal area. We use this as the datapoint for the area-optimized circuit. This step also outputs the total critical path of the circuit.

4. Timing simulations have then been performed on the synthesized netlist. Correctness was again verified against test vectors.

5. For the 90nm and 15nm processes, the switching activity of each gate of the circuit was collected during post-synthesis simulation. The average power was obtained using `Synopsys Design Compiler`, using the back annotated switching activity.

6. To generate latency-optimized circuits, we constrain the total signal delay between the input/output ports to an impossibly low value, like 1ps. The compiler of course fails, but it still outputs a circuit with a critical path as small as it can find.

We coded the S-Boxes as LUTs, except for the `AES`. Synthesis software can often generate better RTL involving 4-bit S-Boxes starting from LUTs than from ad-hoc code (even if, say, hand-optimized or generated with software like PEIGEN). Larger S-Boxes are a different matter, because of their higher intrinsic complexity. For the `AES` we used the Maximov-Ekdahl circuit [ME19]. For `SKINNY-128` and `BipBip` we tried both tables and the circuits from the designers, and the table yielded significantly better area and latency.

6.2 Results

All results are presented in Tables 5 to 7. We include an energy-related *Figure of Merit* (FoM), namely the product of Delay and Power, divided by block size, normalized as a percentage of the `AES`-128 value. *With this definition, lower values are better.* This is just one of the many possible Figures of Merit that can be associated to a circuit: For instance, in [BDN+10], the FoM is defined as throughput over the square of the area.

We are not allowed to report power consumption at 5nm. Hence, we use area times delay over block size as a FoM. Note that power depends not only on area but also on the specific gates used; a cipher with an 8-bit S-Box will use a different mix of gates than a cipher with a 4-bit S-Box, and their power consumption profiles are ignored when just using the area. Hence, the two definitions of FoM cannot be directly compared.

Note that this FoM only applies to the primitive, not to a mode of operation. Since in XEX we need two invocations of the same circuit of a non-tweakable block cipher to encrypt a block, we may want to double the latter's FoM for comparison to a TBC.

Our results prove that `QARMAv2` offers advantages with respect to all other ciphers we tested in fully unrolled implementations. At the 64-bit block level, it is consistently faster than tweaked XEX constructions based on `MIDORI`-64, `PRINCE`, while allowing longer tweaks. At the 128-bit block level, the advantage is even more significant. `QARMAv2`-128 is both smaller and faster than the corresponding `AES` at all security levels, with a Power-Delay-per-bit FoM advantage ranging from 3 to 10. Although `MIDORI`-128 is slightly smaller and faster, it lacks tweakability. `PRESENT`-128 shows mixed results, being sometimes faster or slower, sometimes larger or smaller, but it is also not tweakable. `SPEEDY` is faster, but its security margins need to be re-calculated, its block size is unusual for most applications, and its inverse is slower. `SKINNY` is designed for small round-based implementations, and

Table 5: Comparative Evaluation Metrics for the STM 90nm Process and Library.

		Area optimized					*Latency optimized*				
		Area		Delay	Power	FoM	Area		Delay	Power	FoM
Cipher	Rounds	μm^2	GE	ns	mW	% AES	μm^2	GE	ns	mW	% AES
AES-128	10	392477	89199	42.11	87.12	100	1007463	228969	7.24	220.57	100
AES-192	12	466704	106069	49.54	112.22	151	1174522	266937	8.66	306.05	166
AES-256	14	388858	88377	75.79	151.61	313	1013796	230408	11.86	396.29	294
PRESENT-80	31	65059	14786	24.15	9.93	13.1	204429	46461	4.84	38.05	23.0
PRESENT-128	31	68532	15575	23.63	10.49	13.5	209871	47698	4.87	37.32	22.8
MIDORI-64	16	32531	7394	13.94	2.72	2.06	96172	21857	3.08	12.69	4.90
MIDORI-128	20	85308	19388	18.82	10.09	5.18	284115	64572	4.09	46.78	12.0
PRINCE	12	27898	6340	11.35	1.70	1.06	89210	20275	2.56	12.25	3.92
Orthros	12	98151	22307	10.60	6.57	1.90	299910	68162	2.41	26.06	3.93
SPEEDY-6	6	112098	25477	12.36	9.82	2.20	340428	77370	2.54	26.17	2.77
SPEEDY-7	7	107958	24536	15.02	9.45	2.58	333465	75788	2.95	27.67	3.41
ASCON-p^{12}	12	133855	30421	12.85	5.45	1.91	416723	94710	3.07	17.57	3.38
SKINNY-64-192	32	68966	15674	42.76	15.07	17.6	217367	49402	6.06	48.91	18.6
SKINNY-128-128	40	335075	76153	102.71	148.94	417	966836	219735	12.37	358.79	278
SKINNY-128-384	40	377083	85700	102.72	167.62	470	991088	225247	12.49	376.99	295
MANTIS-7	16	41408	9411	15.66	3.34	2.85	125087	28429	3.26	13.11	5.35
MANTIS-8	18	46443	10555	17.37	4.07	3.85	136300	30977	3.67	16.40	7.54
BIPBIP-Dec	11	91747	20852	14.98	5.05	11.0	177407	40320	2.26	9.25	6.98
BIPBIP-Enc	11	120880	27473	33.47	11.43	55.6	297645	67647	4.79	29.87	47.8
QARMAv1-64-σ_0 ($r = 5$, PAC)	12	32762	7446	12.44	2.21	1.50	112636	25599	2.68	9.93	3.34
QARMAv1-64 ($r = 7$)	16	45023	10232	18.25	4.22	4.12	152671	34698	3.79	19.81	9.40
QARMAv1-128 ($r = 9$)	20	111569	25357	23.01	13.18	8.27	383538	87168	4.81	66.63	20.0
QARMAv1-128 ($r = 11$)	24	121448	27602	25.82	26.90	18.9	396003	90001	5.71	84.10	30.0
QARMAv2-64-σ_0 ($r = 4$)	10	26390	5998	9.12	1.18	0.59	85513	19435	2.11	6.41	1.69
QARMAv2-64-σ_0 ($r = 5$)	12	31670	7198	11.24	1.77	1.08	99035	22508	2.57	8.78	2.83
QARMAv2-64-σ_0 ($r = 6$)	14	36818	8368	13.21	2.43	1.75	117655	26740	2.97	12.22	4.55
QARMAv2-64 ($r = 7$)	16	43857	9968	16.91	3.66	3.37	134419	30550	3.56	16.70	7.45
QARMAv2-64 ($r = 9$)	20	54996	12499	21.14	5.89	6.79	168483	38292	4.44	26.34	14.6
QARMAv2-128-128 ($r = 9$)	20	107474	24426	21.25	11.59	6.71	336499	76477	4.56	54.83	15.7
QARMAv2-128-128 ($r = 11$)	24	126193	28680	24.25	27.32	18.1	372749	84716	5.26	69.01	22.7
QARMAv2-128-192 ($r = 13$)	28	147408	33502	28.28	37.50	28.9	434831	98825	6.14	96.15	37.0
QARMAv2-128-256 ($r = 15$)	32	168622	38323	32.48	49.57	43.9	494968	112493	7.02	125.05	55.0

its performance and area suffer significantly in fully unrolled implementations. On the other hand, although round-based implementations of QARMAv1 and QARMAv2 may not be overtly large or slow, they probably not compare favorably to SKINNY.

For the same value of r, MANTIS-r has the same number of rounds as QARMAv1 and QARMAv2, while lighter and faster. For single block tweaks QARMAv2-64 with $r = 7$ has similar area and latency to MANTIS-7 while likely offering a larger security margin.

For BipBip, our area synthesis results contrast with those in [BDD+23], possibly due to a different GE baseline. BipBip outperforms QARMAv2-64-r-σ_0, even at $r = 5$.

Some recent wide block constructions are not presented in the performance tables, like AESQ [KLMR16] and Pholkos [BLLS22]. They are obviously not lightweight primitives, however comparing them to, say QARMAv2-128, may still be of interest. First, Pholkos supports only 128-bit tweaks, which we argued to be too short in the Introduction. Secondly, the area of Pholkos-128$\,n$ with s steps (one step is two unkeyed AES rounds) is roughly $ns/5$ times and its latency $s/5$ times those of AES-128. The FoMs are thus approximately equal to $(s/5)^2 \times 100\,\%$. Since s is 8 for Pholkos-256 and 10 for Pholkos-512, we have FoMs of 256% and 400%. AESQ has the same FoM as Pholkos-512. The only advantage of these primitives is that they can be accelerated in software using architectural AES instructions. In Appendix F we show how our design philosophy (together with ideas from QARMAv1) can be used to provide lightweight alternatives to AESQ and Pholkos.

Remark 4. The synthesis step occasionally yields unusual outcomes, like the area of AES-256 being smaller than AES-192 in the 90nm process. Ideally, the FoMs of AES-192 and AES-256 should be approximately $(12/10)^2 = 144\,\%$ and $(14/10)^2 = 196\,\%$. This simply means that the synthesis step selected a circuit which it deemed best according to an internal objective function among the wide – but not comprehensive – space of *explored* options.

Table 6: Comparative Evaluation Metrics for the Nangate 15nm Process and Library.

		Area optimized					*Latency optimized*				
		Area		Delay	Power	FoM	Area		Delay	Power	FoM
Cipher	Rounds	μm^2	GE	ps	mW	% AES	μm^2	GE	ps	mW	% AES
AES-128	10	22478	114329	1728	26.63	100	28071	142777	1026	31.14	100
AES-192	12	26698	135796	2061	34.96	157	34025	173063	1221	36.39	139
AES-256	14	31744	161459	2319	47.64	240	41601	211594	1422	49.12	219
PRESENT-80	31	3808	19370	1001	2.87	11.2	6261	31846	711	4.63	20.2
PRESENT-128	31	3953	20105	960	2.67	12.5	6446	32787	706	4.56	20.6
MIDORI-64	16	1929	9814	629	0.96	2.61	2567	13054	455	1.31	3.76
MIDORI-128	20	5102	25952	851	3.35	6.20	6976	35484	604	4.47	8.44
PRINCE	12	1664	8465	536	0.67	1.56	2338	11891	372	0.89	2.08
Orthros	12	5766	29329	486	2.33	2.46	7439	37838	352	2.64	2.90
SPEEDY-6	6	5220	26551	545	2.06	1.63	7539	38346	355	2.37	1.76
SPEEDY-7	7	6610	33618	659	3.18	3.02	8875	45139	419	3.51	3.06
ASCON-p^{12}	12	8562	43545	568	2.07	1.63	10408	52939	436	1.82	2.48
SKINNY-64-192	32	4069	20693	1953	5.03	42.6	6162	31332	892	5.30	29.6
SKINNY-128-128	40	18520	94200	3862	40.89	343	29643	150777	1763	49.47	196
SKINNY-128-384	40	20480	104172	4098	45.18	402	30896	157148	1751	47.32	186
MANTIS-7	16	2412	12268	723	1.23	3.87	3364	17110	474	1.32	3.92
MANTIS-8	18	2703	13749	814	1.56	5.52	3759	19121	529	1.66	5.50
BIPBIP-Dec	11	5344	27180	710	2.01	16.5	6554	33843	327	1.42	7.75
BIPBIP-Enc	11	7031	35760	1520	3.99	70.3	10334	52561	683	3.86	44.0
QARMAv1-64-σ_0 ($r = 5$, PAC)	12	1879	9555	550	0.72	1.10	2771	14091	354	0.92	2.02
QARMAv1-64 ($r = 7$)	16	2688	13672	828	1.36	3.12	3863	19649	513	1.67	5.36
QARMAv1-128 ($r = 9$)	20	6465	32889	1042	5.73	13.0	9681	49236	649	5.50	11.2
QARMAv1-128 ($r = 11$)	24	7765	39493	1244	6.87	18.6	11633	59169	780	8.17	19.9
QARMAv2-64-σ_0 ($r = 4$)	10	1583	8053	447	0.48	0.93	2381	12110	305	0.59	1.13
QARMAv2-64-σ_0 ($r = 5$)	12	1897	9650	530	0.70	1.61	2839	14439	371	0.90	2.09
QARMAv2-64-σ_0 ($r = 6$)	14	2202	11201	620	0.95	2.56	3349	17031	426	1.24	3.30
QARMAv2-64 ($r = 7$)	16	2580	13122	752	1.36	4.45	3754	19095	497	1.59	4.95
QARMAv2-64 ($r = 9$)	20	3231	16887	952	2.13	8.74	4738	24100	618	2.58	9.98
QARMAv2-128-128 ($r = 9$)	20	6336	32226	909	4.39	8.67	9292	47261	623	4.96	9.67
QARMAv2-128-128 ($r = 11$)	24	7585	38580	1140	6.24	15.5	11142	56674	749	7.19	16.9
QARMAv2-128-192 ($r = 13$)	28	8842	44975	1330	8.47	24.5	12991	66074	877	10.09	27.7
QARMAv2-128-256 ($r = 15$)	32	10107	51411	1516	11.08	36.5	14734	74941	1002	13.00	40.8

Remark 5. An important remark concerns the measurement of circuit area as Gate Equivalence (GE) i.e. relative to the area of a NAND gate. Each gate exists in several versions, differing in arity, fanout, and voltage, and in the literature it is uncommon to find which version of NAND2 is used as the baseline. Therefore, GE comparisons hold only with the same process and (version of the) library. Comparison across different papers do not necessarily apply, unless they are using the exact same tools. As a results, area-optimized SKINNY-128-128 ($r = 40$) has GE values of 76153, 94200, and 49553 in the three processes, which are higher than the value 32415 in [BJK+16].

7 Conclusions and Open Questions

We have introduced the tweakable block cipher family QARMAv2, an update to the QARMAv1 design. Like its predecessor, QARMAv2 targets low-latency fully unrolled implementations for memory encryption and constructing timing critical MACs. However, it is clearly suitable for general purpose use, except possibly when the smallest possible round-based implementation is required. Two significant enhancements in the new cipher include the accommodation of longer tweaks and the shift away from basing security levels on time/data tradeoffs. All the building blocks and their composition have been extensively analyzed. We took a cautious stance, opting to keep solid security margins rather than decreasing the number of rounds or employing a lighter S-Box. Despite this, QARMAv2 is a competitive lightweight design, as confirmed by our hardware implementations.

Some techniques used in this paper may be of independent interest, such as: our MILP models for diffusion matrices; a comparative analysis of a full reflection cipher versus an iterative half-cipher; our boomerang attack framework that includes the reflector; and our

Table 7: Comparative Evaluation Metrics for the TSMC 5nm Process and Library.

		Area optimized				*Latency optimized*			
		Area		Delay	FoM	Area		Delay	FoM
Cipher	Rounds	μm^2	GE	ps	% AES	μm^2	GE	ps	% AES
AES-128	10	2304.1	28873	3064	100	4520.6	56648	1791	100
AES-192	12	2635.4	33025	3686	138	5023.6	62952	2153	134
AES-256	14	3238.7	40585	4290	197	6191.5	77587	2513	192
PRESENT-80	31	812.0	10175	1836	42.2	1815.7	22752	953	42.8
PRESENT-128	31	848.8	10636	1841	44.2	1824.1	22858	958	43.2
MIDORI-64	16	443.5	5557	921	10.6	761.8	9546	678	12.7
MIDORI-128	20	1085.1	13597	1156	17.8	1954.5	24492	840	40.6
PRINCE	12	334.6	4193	710	6.74	672.1	8422	534	8.86
Orthros	12	1262.5	15820	681	12.2	2284.3	28624	497	14.0
SPEEDY-6	6	1795.5	22499	787	13.3	3133.8	39270	468	12.1
SPEEDY-7	7	2109.2	26431	924	18.4	3599.6	45107	552	16.2
ASCON-p^{12}	12	2228.3	27923	826	26.1	2766.8	34671	507	17.3
SKINNY-64-192	32	918.3	11507	1951	25.4	1682.0	21078	1254	26.0
SKINNY-128-128	40	3986.3	49953	4371	247	9241.0	115800	2164	247
SKINNY-128-384	40	4513.6	56560	4348	278	9527.5	119391	2177	256
MANTIS-7	16	485.6	6085	854	11.8	788.4	9879	683	13.3
MANTIS-8	18	545.8	6839	974	15.1	861.0	10789	774	16.5
BIPBIP-Dec	11	303.7	3806	647	14.8	381.1	4776	436	11.0
BIPBIP-Enc	11	514.7	6450	1480	57.6	1090.3	13662	909	65.3
QARMAv1-64-σ_0 ($r = 5$, PAC)	12	394.7	4946	728	8.14	707.0	8860	525	9.16
QARMAv1-64 ($r = 7$)	16	551.7	6913	1030	16.1	996.6	12489	731	18.0
QARMAv1-128 ($r = 9$)	20	1422.3	17823	1290	26	2535.8	31776	912	28.6
QARMAv1-128 ($r = 11$)	24	1635.6	20496	1561	36.2	3078.3	38575	1091	41.5
QARMAv2-64-σ_0 ($r = 4$)	10	309.7	3881	606	5.32	495.9	6214	430	5.27
QARMAv2-64-σ_0 ($r = 5$)	12	374.6	4694	721	7.65	600.8	7529	514	7.63
QARMAv2-64-σ_0 ($r = 6$)	14	435.4	5456	829	10.2	721.2	9038	600	10.7
QARMAv2-64 ($r = 7$)	16	537.0	6729	936	14.2	954.4	11959	706	16.6
QARMAv2-64 ($r = 9$)	20	675.2	8461	1173	22.4	1187.3	14879	885	26.0
QARMAv2-128-128 ($r = 9$)	20	1347.5	16886	1170	22.3	2337.5	29292	890	25.7
QARMAv2-128-128 ($r = 11$)	24	1620.3	20305	1409	32.3	2875.8	36037	1068	37.9
QARMAv2-128-192 ($r = 13$)	28	1893.5	23727	1645	44.1	3333.0	41778	1248	51.4
QARMAv2-128-256 ($r = 15$)	32	2166.8	27152	1879	57.7	3797.8	47592	1425	66.8

approach to block size doubling, which goes to greater length than the approaches used for AESQ and Pholkos to re-use the cryptographic properties of the base design.

Regarding the comparison between purely iterative ciphers and reflection ciphers, our Boomerang cryptanalysis (Section 4.2) shows that lopsided distinguishers exhibit greater reach or higher probability than symmetric ones. This echoes findings in the cryptanalysis of QARMAv1 (Appendix G), where asymmetric attacks target more rounds. Additionally, in reflector constructions, the faster ramp-up of the number of active cells in linear trails and related-tweak differential characteristics suggests higher security compared to pure iterative designs. However, the lesson from Remark 2 in Section 2.7 cautions that any added structure may come with its own pitfalls.

In Section 4.7 we proved that certain forms of subspace trail analysis do not threaten the security of QARMAv2, regardless of the round constants. In fact, the constants play no role at all in our proof. While the constants harden against slide attacks, the mix of different round types may suffice. We kept the round constants dense due to the design philosophy inherited from QARMAv1, i.e. a conservative, "defense-in-depth" approach. This said, we could even envision a "QARMA$^{\texttt{zero}}$" version of QARMAv2 with the round constants set to 0, except for α and β. If secure, it would be an enticing alternative for software and round-based hardware implementations. Hence, properly determining the security margins of such a variant is an interesting problem in its own right.

Our research raises some intriguing questions: Can the cycle structure of an S-Box as a permutation be linked to its complexity (e.g., multiplicative complexity, depth, or area)? Can we find more effective tweak schedules that still admit a compact description?

Acknowledgements

We express our gratitude to: Arm Limited, for generously funding this research and sponsoring cloud computing time and hardware; Gary Gorman for providing the implementation data on the TSMC 5nm process; Andreas Sandberg for many interesting conversations on cryptographic memory protection; Tim Beyne for a very useful discussion on nonlinear invariants of round functions; 包珍珍 for assistance with PEIGEN; 伊藤竜馬 for painstakingly double checking our test vectors; and Gurobi Optimization, LLC, for generously providing research licenses of their MILP Solver. Last, but not least, we want to thank the reviewers and the shepherd for outstandingly fulfilling their duties.

References

[ABI+18] Gianira N. Alfarano, Christof Beierle, Takanori Isobe, Stefan Kölbl, and Gregor Leander. ShiftRows alternatives for AES-like ciphers and optimal cell permutations for Midori and Skinny. *IACR Trans. Symmetric Cryptol.*, 2018(2):20–47, 2018. doi:10.13154/tosc.v2018.i2.20-47. Cited on page 10.

[ADG+19] Ralph Ankele, Christoph Dobraunig, Jian Guo, Eran Lambooij, Gregor Leander, and Yosuke Todo. Zero-Correlation attacks on tweakable block ciphers with linear tweakey expansion. *IACR Trans. Symmetric Cryptol.*, 2019(1):192–235, 2019. doi:10.13154/tosc.v2019.i1.192-235. Cited on page 49.

[AMS+22] 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 Cryptology ePrint Archive, Paper 2022/1472*, 2022. URL: https://eprint.iacr.org/2022/1472. Cited on pages 1, 2, and 3.

[Arm16] Arm. ARMv8-A architecture – 2016 additions. ARM Connected blog. https://www.community.arm.com/processors/b/blog/posts/armv8-a-architecture-2016-additions, October 2016. Cited on page 1.

[Ava17] 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. doi:10.13154/tosc.v2017.i1.4-44. Cited on pages 1, 14, 22, 23, 24, 47, 49, and 51.

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 411–436. Springer, 2015. doi:10.1007/978-3-662-48800-3_17. Cited on page 10.

[BCF+22] Ritam Bhaumik, André Chailloux, Paul Frixons, Bart Mennink, and María Naya-Plasencia. Block cipher doubling for a post-quantum world. *IACR Cryptology ePrint Archive, Paper 2022/1342*, 2022. URL: https://eprint.iacr.org/2022/1342. Cited on page 13.

[BCFG+21] Marek Broll, Federico Canale, Antonio Flórez-Gutiérrez, Gregor Leander, and María Naya-Plasencia. Generic framework for key-guessing improvements. In *ASIACRYPT 2021*, volume 13090 of *LNCS*, pages 453–483. Springer, 2021. doi:10.1007/978-3-030-92062-3_16. Cited on page 20.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - Extended abstract. In *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 208–225. Springer, 2012. doi:10.1007/978-3-642-34961-4_14. Cited on page 1.

[BDD+23] Yanis Belkheyar, Joan Daemen, Christoph Dobraunig, Santosh Ghosh, and Shahram Rasoolzadeh. BipBip: A low-latency tweakable block cipher with small dimensions. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):326–368, 2023. doi:10.46586/tches.v2023.i1.326-368. Cited on pages 9, 26, and 28.

[BDN+10] Stéphane Badel, Nilay Dagtekin, Jorge Nakahara, Khaled Ouafi, Nicolas Reffé, Pouyan Sepehrdad, Petr Susil, and Serge Vaudenay. ARMADILLO: A multi-purpose cryptographic primitive dedicated to hardware. In *CHES 2010*, volume 6225 of *LNCS*, pages 398–412. Springer, 2010. doi:10.1007/978-3-642-15031-9_27. Cited on page 27.

[BEK+20] Dusan Bozilov, Maria Eichlseder, Miroslav Knezevic, Baptiste Lambin, Gregor Leander, Thorben Moos, Ventzislav Nikov, Shahram Rasoolzadeh, Yosuke Todo, and Friedrich Wiemer. PRINCEv2 - More security for (almost) no overhead. In *SAC 2020*, volume 12804 of *LNCS*, pages 483–511. Springer, 2020. doi:10.1007/978-3-030-81652-0_19. Cited on pages 1 and 7.

[Bey18] Tim Beyne. Block cipher invariants as eigenvectors of correlation matrices. In *ASIACRYPT 2018*, volume 11272 of *LNCS*, pages 3–31. Springer, 2018. doi:10.1007/978-3-030-03326-2_1. Cited on page 14.

[Bey23] Tim Beyne. An invariant of the round function of QARMAv2-64. *IACR Cryptology ePrint Archive, Paper 2023/963*, 2023. URL: https://eprint.iacr.org/2023/963. Cited on pages 14 and 52.

[BGGS20] Zhenzhen Bao, Chun Guo, Jian Guo, and Ling Song. TNT: How to tweak a block cipher. In *EUROCRYPT 2020*, volume 12106 of *LNCS*, pages 641–673. Springer, 2020. doi:10.1007/978-3-030-45724-2_22. Cited on page 2.

[BI99] Mihir Bellare and Russell Impagliazzo. A tool for obtaining tighter security analyses of pseudorandom function based constructions, with applications to PRP to PRF conversion. *IACR Cryptology ePrint Archive, Paper 1999/024*, 1999. URL: http://eprint.iacr.org/1999/024. Cited on page 24.

[BIL+21] Subhadeep Banik, Takanori Isobe, Fukang Liu, Kazuhiko Minematsu, and Kosei Sakamoto. Orthros: A low-latency PRF. *IACR Trans. Symmetric Cryptol.*, 2021(1):37–77, 2021. doi:10.46586/tosc.v2021.i1.37-77. Cited on page 26.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *CRYPTO 2016*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016. doi:10.1007/978-3-662-53008-5_5. Cited on pages 1, 10, 26, and 29.

[BLLS22] Jannis Bossert, Eik List, Stefan Lucks, and Sebastian Schmitz. Pholkos - Efficient large-state tweakable block ciphers from the AES round function. In Steven D. Galbraith, editor, *CT-RSA 2022*, volume 13161 of *LNCS*, pages 511–536. Springer, 2022. doi:10.1007/978-3-030-95312-6_21. Cited on pages 2, 3, 13, and 28.

[BW99] Alex Biryukov and David Wagner. Slide attacks. In *FSE '99*, volume 1636 of *LNCS*, pages 245–259. Springer, 1999. doi:10.1007/3-540-48519-8_18. Cited on page 21.

[CHP+18] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang connectivity table: A new cryptanalysis tool. In *EUROCRYPT 2018*, volume 10821 of *LNCS*, pages 683–714. Springer, 2018. doi:10.1007/978-3-319-78375-822. Cited on page 16.

[CK08] Jiali Choy and Khoongming Khoo. New applications of differential bounds of the SDS structure. In *Information Security, 11th International Conference, ISC 2008, Taipei, Taiwan, September 15-18, 2008. Proceedings*, pages 367–384, 2008. Corrected version at http://eprint.iacr.org/2008/395. URL: http://dx.doi.org/10.1007/978-3-540-85886-7_26, doi:10.1007/978-3-540-85886-7_26. Cited on page 50.

[CP02] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 267–287. Springer, 2002. doi:10.1007/3-540-36178-2_17. Cited on page 23.

[CSQ07] Baudoin Collard, François-Xavier Standaert, and Jean-Jacques Quisquater. Improving the time complexity of Matsui's linear cryptanalysis. In *ICISC 2007*, volume 4817 of *LNCS*, pages 77–88. Springer, 2007. doi:10.1007/978-3-540-76788-6_7. Cited on page 20.

[DEKM16] Christoph Dobraunig, Maria Eichlseder, Daniel Kales, and Florian Mendel. Practical key-recovery attack on MANTIS5. *IACR Trans. Symmetric Cryptol.*, 2016(2):248–260, 2016. doi:10.13154/tosc.v2016.i2.248-260. Cited on page 18.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. doi:10.1007/978-3-662-04722-4. Cited on page 49.

[DWLW22] Juan Du, Wei Wang, Muzhou Li, and Meiqin Wang. Related-tweakey impossible differential attack on QARMA-128. *Sci. China Inf. Sci.*, 65(2), 2022. doi:10.1007/s11432-019-2628-9. Cited on page 49.

[EK18] Maria Eichlseder and Daniel Kales. Clustering related-tweak characteristics: Application to MANTIS-6. *IACR Trans. Symmetric Cryptol.*, 2018(2):111–132, 2018. doi:10.13154/tosc.v2018.i2.111-132. Cited on pages 18 and 26.

[EM91] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. In *ASIACRYPT '91*, volume 739 of *LNCS*, pages 210–224. Springer, 1991. doi:10.1007/3-540-57332-1_17. Cited on page 24.

[FGNP20] Antonio Flórez-Gutiérrez and María Naya-Plasencia. Improving key-recovery in linear attacks: Application to 28-round PRESENT. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 221–249. Springer, 2020. doi:10.1007/978-3-030-45721-1_9. Cited on page 20.

[GRR16] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. Subspace trail cryptanalysis and its applications to AES. *IACR Trans. Symmetric Cryptol.*, 2016(2):192–225, 2016. doi:10.13154/tosc.v2016.i2.192-225. Cited on page 22.

[Gue16] Shay Gueron. A memory encryption engine suitable for general purpose processors. *IACR Cryptology ePrint Archive, Paper 2016/204*, 2016. URL: http://eprint.iacr.org/2016/204. Cited on pages 1 and 25.

[HI19] Akinori Hosoyamada and Tetsu Iwata. 4-Round Luby-Rackoff construction is a qPRP. In *ASIACRYPT 2019*, volume 11921 of *LNCS*, pages 145–174. Springer, 2019. doi:10.1007/978-3-030-34578-5_6. Cited on page 13.

[HNE22] Hosein Hadipour, Marcel Nageler, and Maria Eichlseder. Throwing boomerangs into Feistel structures: Application to CLEFIA, WARP, LBlock, LBlock-s and TWINE. *IACR Trans. Symmetric Cryptol.*, 2022(3):271–302, 2022. doi:10.46586/tosc.v2022.i3.271-302. Cited on page 18.

[HSE23] Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In *EUROCRYPT 2023*, volume 14007 of *LNCS*, pages 128–157. Springer, 2023. doi:10.1007/978-3-031-30634-1_5. Cited on page 20.

[HT13] Michael Henson and Stephen Taylor. Memory encryption: A survey of existing techniques. *ACM Comput. Surv.*, 46(4):53:1–53:26, 2013. doi:10.1145/2566673. Cited on page 1.

[IEE18] IEEE approved draft standard for cryptographic protection of data on block-oriented storage devices. *P1619/D28, August 2018*, pages 1–38, 2018. Cited on page 1.

[IKMP19] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Romulus v1 – A submission to NIST lightweight cryptography project, 2019. Cited on page 26.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the titans: The Romulus and Remus families of lightweight AEAD algorithms. *IACR Trans. Symmetric Cryptol.*, 2020(1):43–120, 2020. doi:10.13154/tosc.v2020.i1.43-120. Cited on page 26.

[Int21] Intel. Intel® TDX module 1.5 base architecture specification, September 2021. URL: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-trust-domain-extensions.html. Cited on pages 1 and 25.

[JLK+23] Jonas Juffinger, Lukas Lamster, Andreas Kogler, Moritz Lipp, Maria Eichlseder, and Daniel Gruss. CSI:RowHammer – Cryptographic security and integrity against RowHammer. In *S&P'23*, 2023. Cited on page 24.

[JLM+17] Ashwin Jha, Eik List, Kazuhiko Minematsu, Sweta Mishra, and Mridul Nandi. XHX - A framework for optimally secure tweakable block ciphers from classical block ciphers and universal hashing. In *LATINCRYPT 2017*, volume 11368 of *LNCS*, pages 207–227. Springer, 2017. doi:10.1007/978-3-030-25283-0_12. Cited on page 2.

[JN20] Ashwin Jha and Mridul Nandi. Tight security of cascaded LRW2. *Journal of Cryptology*, 33(3):1272–1317, 2020. doi:10.1007/s00145-020-09347-y. Cited on page 2.

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In *ASIACRYPT 2014*, volume 8874 of *LNCS*, pages 274–288. Springer, 2014. doi:10.1007/978-3-662-45608-8_15. Cited on page 2.

[JNRV20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing Grover oracles for quantum key search on AES and LowMC. In *EUROCRYPT 2020*, volume 12106 of *LNCS*, pages 280–310. Springer, 2020. doi:10.1007/978-3-030-45724-2_10. Cited on page 3.

[JV04] Pascal Junod and Serge Vaudenay. Perfect diffusion primitives for block ciphers. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 84–99. Springer, 2004. doi:10.1007/978-3-540-30564-4_6. Cited on page 50.

[Kar07] Orhun Kara. Reflection attacks on product ciphers. *IACR Cryptology ePrint Archive, Paper 2007/043*, 2007. URL: http://eprint.iacr.org/2007/043. Cited on page 21.

[Kar08] Orhun Kara. Reflection Cryptanalysis of Some Ciphers. In *INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 294–307. Springer, 2008. doi:10.1007/978-3-540-89754-5_23. Cited on page 24.

[KLMR16] Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. Haraka v2 - efficient short-input hashing for post-quantum applications. *IACR Trans. Symmetric Cryptol.*, 2016(2):1–29, 2016. doi:10.13154/tosc.v2016.i2.1-29. Cited on pages 2, 3, 13, and 28.

[KPW16] David Kaplan, Jeremy Powell, and Tom Woller. AMD memory encryption white paper, April 2016. URL: http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf. Cited on page 1.

[LHW19] Muzhou Li, Kai Hu, and Meiqin Wang. Related-tweak statistical saturation cryptanalysis and its application on QARMA. *IACR Trans. Symmetric Cryptol.*, 2019(1):236–263, 2019. doi:10.13154/tosc.v2019.i1.236-263. Cited on page 49.

[LJ18] Rongjia Li and Chenhui Jin. Meet-in-the-Middle attacks on reduced-round QARMA-64/128. *Comput. J.*, 61(8):1158–1165, 2018. doi:10.1093/comjnl/bxy045. Cited on page 49.

[LMR15] Gregor Leander, Brice Minaud, and Sondre Rønjom. A generic approach to invariant subspace attacks: Cryptanalysis of Robin, iSCREAM and Zorro. In *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 254–283. Springer, 2015. doi:10.1007/978-3-662-46800-5_11. Cited on page 22.

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, 2002. doi:10.1007/3-540-45708-9. Cited on page 1.

[LRW11] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. *Journal of Cryptology*, 24(3):588–613, 2011. doi:10.1007/s00145-010-9073-y. Cited on page 2.

[LST12] Will Landecker, Thomas Shrimpton, and R. Seth Terashima. Tweakable blockciphers with beyond birthday-bound security. In *CRYPTO 2012*, volume 7417 of *LNCS*, pages 14–30. Springer, 2012. doi:10.1007/978-3-642-32009-5_2. Cited on page 2.

[LZG+20] Ya Liu, Tiande Zang, Dawu Gu, Fengyu Zhao, Wei Li, and Zhiqiang Liu. Improved cryptanalysis of reduced-version QARMA-64/128. *IEEE Access*, 8:8361–8370, 2020. doi:10.1109/ACCESS.2020.2964259. Cited on pages 22 and 49.

[Mat94] Mitsuru Matsui. On correlation between the order of S-boxes and the strength of DES. In *EUROCRYPT '94*, volume 950 of *LNCS*, pages 366–375. Springer, 1994. doi:10.1007/BFb0053451. Cited on page 10.

[ME19] Alexander Maximov and Patrik Ekdahl. New circuit minimization techniques for smaller and faster AES SBoxes. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(4):91–125, 2019. doi:10.13154/tches.v2019.i4.91-125. Cited on page 27.

[Men15] Bart Mennink. Optimally secure tweakable blockciphers. In *FSE 2015*, volume 9054 of *LNCS*, pages 428–448. Springer, 2015. doi:10.1007/978-3-662-48116-5_21. Cited on page 2.

[Men18] Bart Mennink. Towards tight security of cascaded LRW2. In *TCC 2018*, volume 11240 of *LNCS*, pages 192–222. Springer, 2018. doi:10.1007/978-3-030-03810-6_8. Cited on page 2.

[MMR+15] Mayler Martins, Jody Maick Matos, Renato P. Ribas, Andr'e Reis, Guilherme Schlinker, Lucio Rech, and Jens Michelsen. Open cell library in 15nm freepdk technology. In *ISPD 2015*, page 171–178, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2717764.2717783. Cited on page 26.

[MPS+21] Dominic P. Mulligan, Gustavo Petri, Nick Spinale, Gareth Stockwell, and Hugo J. M. Vincent. Confidential computing - a brave new world. In *SEED 2021*, pages 132–138. IEEE, 2021. doi:10.1109/SEED51797.2021.00025. Cited on page 1.

[MV04] David A. McGrew and John Viega. The security and performance of the Galois/counter mode (GCM) of operation. In *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, 2004. doi:10.1007/978-3-540-30556-9_27. Cited on page 1.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Inscrypt 2011*, volume 7537 of *LNCS*, pages 57–76. Springer, 2011. doi:10.1007/978-3-642-34704-7_5. Cited on page 10.

[NIS] NIST. Lightweight cryptography. URL: https://csrc.nist.gov/projects/lightweight-cryptography. Cited on page 15.

[NIS01] NIST. Recommendation for block cipher modes of operation: Methods and techniques, special publication 800-38a. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, United States, December 2001. doi:10.6028/NIST.SP.800-38A. Cited on page 2.

[NIS18] NIST. Submission requirements and evaluation criteria for the lightweight cryptography standardization process, 2018. URL: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf. Cited on page 15.

[NIS23] NIST. Decision to revise NIST SP 800-38a, Recommendation for block cipher modes of operation: Methods and techniques, April 2023. URL: https://csrc.nist.gov/News/2023/decision-to-revise-nist-sp-800-38a. Cited on pages 2 and 48.

[Qua17] Qualcomm Product Security. Pointer authentication on ARMv8.3. design and analysis of the new software security instructions. https://www.qualcomm.com/documents/whitepaper-pointer-authentication-armv83, January 2017. Cited on page 1.

[Rog04] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, 2004. doi:10.1007/978-3-540-30539-2_2. Cited on pages 1 and 2.

[SBY+15] Hadi Soleimany, Céline Blondeau, Xiaoli Yu, Wenling Wu, Kaisa Nyberg, Huiling Zhang, Lei Zhang, and Yanfeng Wang. Reflection cryptanalysis of PRINCE-like ciphers. *Journal of Cryptology*, 28(3):718–744, 2015. URL: http://dx.doi.org/10.1007/s00145-013-9175-4, doi:10.1007/s00145-013-9175-4. Cited on page 24.

[SII23] Kosei Sakamoto, Ryoma Ito, and Takanori Isobe. Parallel SAT framework to find clustering of differential characteristics and its applications. Cryptology ePrint Archive, Paper 2023/1227, 2023. https://eprint.iacr.org/2023/1227. URL: https://eprint.iacr.org/2023/1227. Cited on page 49.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In *SAT 2009*, volume 5584 of *LNCS*, pages 244–257. Springer, 2009. doi:10.1007/978-3-642-02777-2_24. Cited on pages 10 and 21.

[SV94] Claus-Peter Schnorr and Serge Vaudenay. Black box cryptanalysis of hash networks based on multipermutations. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 47–57. Springer, 1994. doi:10.1007/BFb0053423. Cited on page 50.

[SWW17] Ling Sun, Wei Wang, and Meiqin Wang. Automatic search of bit-based division property for ARX ciphers and word-based division property. In *ASIACRYPT 2017*, volume 10624 of *LNCS*, pages 128–157. Springer, 2017. doi:10.1007/978-3-319-70694-8_5. Cited on page 21.

[TM16] Yosuke Todo and Masakatu Morii. Bit-based division property and application to Simon family. In *FSE 2016*, volume 9783 of *LNCS*, pages 357–377. Springer, 2016. doi:10.1007/978-3-662-52993-5_18. Cited on page 21.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015. doi:10.1007/978-3-662-46800-5_12. Cited on page 21.

[Tod17] Yosuke Todo. Integral cryptanalysis on full MISTY1. *Journal of Cryptology*, 30(3):920–959, 2017. doi:10.1007/s00145-016-9240-x. Cited on page 21.

[Vau94] Serge Vaudenay. On the need for multipermutations: Cryptanalysis of MD4 and SAFER. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 1994. doi:10.1007/3-540-60590-8_22. Cited on page 50.

[WRP20] Yongzhuang Wei, René Rodríguez, and Enes Pasalic. Cycle structure of generalized and closed loop invariants. *IACR Cryptology ePrint Archive, Paper 2020/1095*, 2020. URL: https://eprint.iacr.org/2020/1095. Cited on page 14.

[WYWP18] Yongzhuang Wei, Tao Ye, Wenling Wu, and Enes Pasalic. Generalized nonlinear invariant attack and a new design criterion for round constants. *IACR Trans. Symmetric Cryptol.*, 2018(4):62–79, 2018. doi:10.13154/tosc.v2018.i4.62-79. Cited on page 14.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 648–678, 2016. doi:10.1007/978-3-662-53887-6_24. Cited on page 21.

[YQC18] Dong Yang, Wen-Feng Qi, and Hua-Jin Chen. Impossible differential attack on QARMA family of block ciphers. Cryptology ePrint Archive, Paper 2018/334, 2018. https://eprint.iacr.org/2018/334. Cited on page 49.

[ZD16] Rui Zong and Xiaoyang Dong. Meet-in-the-middle attack on QARMA block cipher. IACR Cryptology ePrint Archive, Paper 2016/1160, 2016. http://eprint.iacr.org/2016/1160. Cited on pages 22 and 49.

[ZD19] Rui Zong and Xiaoyang Dong. MILP-aided related-tweak/key impossible differential attack and its applications to QARMA, Joltik-BC. *IEEE Access*, 7:153683–153693, 2019. doi:10.1109/ACCESS.2019.2946638. Cited on page 49.

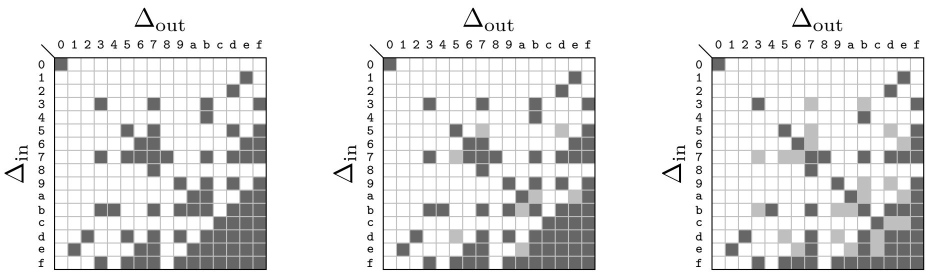
[ZDW18] Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. MILP-aided related-tweak/key impossible differential attack and its applications to QARMA, Joltik-BC. Cryptology ePrint Archive, Paper 2018/142, 2018. https://eprint.iacr.org/2018/142. Cited on page 49.

Appendices

A MILP Modeling

A.1 Modeling `MixColumns`

In this section, we only consider the action of a matrix on a single column of the state, viewed as a vector. Let x_i, resp. y_i for $0 \leq i < 4$ be the cells in a column vector, resp. the cells in the image of said column under `MixColumns`. The weight of a vector is the number of active cells in it. The weight of a transition $\mathsf{y} = M \cdot \mathsf{x}$ is the sum of the weights of x and y. In Figure 9 we show the admissible transitions of Class I and II matrices. We now set to turn them into MILP models.



(a) `XOR` model. **(b)** `QARMA` Class II. **(c)** `QARMA` Class I.

Figure 9: Differential properties of `MixColumns`.

A.1.1 Relations for both Class I and Class II Matrices

To model the fact that *Class I and Class II matrices have branch number 4*, including that an inactive input vector cannot map to an active output vector and vice versa, we introduce an integer auxiliary variable d and add relations

$$\mathsf{x}_i\,,\ \mathsf{y}_i \leq d \quad \text{for} \quad 0 \leq i < 4\ , \quad \text{and} \quad \sum_i \mathsf{x}_i + \sum_i \mathsf{y}_i \geq 4\,d\ , \quad \sum_i \mathsf{x}_i \geq d \quad \text{and} \quad \sum_i \mathsf{y}_i \geq d\ . \tag{15}$$

The simple XOR model of `MixColumns` is given by the following relations for $0 \leq i < 4$:

$$\begin{aligned} \mathsf{y}_i + \sum_{j \neq i} \mathsf{x}_j \geq 2\,\mathsf{y}_i \quad &\text{and} \quad \mathsf{y}_i + \sum_{j \neq i} \mathsf{x}_j \geq 2\,\mathsf{x}_{j'} \quad \text{for all } j' \neq i \\ \mathsf{x}_i + \sum_{j \neq i} \mathsf{y}_j \geq 2\,\mathsf{x}_i \quad &\text{and} \quad \mathsf{x}_i + \sum_{j \neq i} \mathsf{y}_j \geq 2\,\mathsf{y}_{j'} \quad \text{for all } j' \neq i\ . \end{aligned} \tag{16}$$

If the input cell in row k is active, then at least one of the three output cells in the rows $\neq k$ is active (and the reverse direction is also true). This can be expressed as:

$$\sum_{i\neq k} \mathsf{y}_i \geq \mathsf{x}_k \quad \text{and} \quad \sum_{i\neq k} \mathsf{x}_i \geq \mathsf{y}_k \quad \text{for } 0 \leq k < 4 \ . \tag{17}$$

A further property, that follows directly from the XOR model, is: *For an active input cell in row i, and any output cell in a different row j, at least one among that output cell and the input cells in rows $\neq i, j$ must be active.* In equations:

$$\mathsf{y}_j + \sum_{k\neq i,j} \mathsf{x}_k \geq \mathsf{x}_i \quad \text{and} \quad \mathsf{x}_j + \sum_{k\neq i,j} \mathsf{y}_k \geq \mathsf{y}_i \quad \text{for all } i \neq j \ , \ 0 \leq i,j < 4 \ . \tag{18}$$

A.1.2 Class I Specific Relations

It can also easily be seen that a Class I matrix satisfies

$$\mathsf{x}_i + \mathsf{x}_j \geq \mathsf{y}_i - \mathsf{y}_j \quad \text{and} \quad \mathsf{y}_i + \mathsf{y}_i \geq \mathsf{x}_j - \mathsf{x}_j \quad \text{for all } i \neq j \ , \ 0 \leq i,j < 4 \ , \tag{19}$$

i.e. *For any two rows i, j, if output cell i is active and j is not, then at least one of the input cells in rows i and j must be active.*

We can even express the fact that *Class I matrices have no weight five transitions* in a very compact way. To do this, we introduce an integer auxiliary variable d_i for each i and the constraints:

$$\sum_i \mathsf{x}_i + \sum_i \mathsf{y}_i \leq 4 + 4\,d_i \quad \text{and} \quad \sum_i \mathsf{x}_i + \sum_i \mathsf{y}_i \geq 6\,d_i \quad \text{for } 0 \leq i < 4 \ . \tag{20}$$

Indeed, if the total weight of an active transition is not four, then it must be at least five and $d \geq 1$ by the first relation, and by the second relation the total weight is at least six. If the total weight of the transition is four, then d must be 0 by the second relation.

Weight two columns only map to columns with the same active cells or to fully active columns. This is achieved by forcing the same cells to be active in this case and the fact that weight five transitions are not possible (cf. Relation (20)). Let d_{ij}, d'_{ij} be distinct integer auxiliary variables. Then, for all i, j with $0 \leq i < j < 4$, the following relations must be satisfied:

$$\begin{aligned} d_{ij} &\geq \mathsf{x}_i + \mathsf{x}_j - \sum_{k\neq i,j} \mathsf{x}_k - 1 \ , \quad \mathsf{y}_i \geq d_{ij} \quad \text{and} \quad \mathsf{y}_j \geq d_{ij} \\ d'_{ij} &\geq \mathsf{y}_i + \mathsf{y}_j - \sum_{k\neq i,j} \mathsf{y}_k - 1 \ , \quad \mathsf{x}_i \geq d'_{ij} \quad \text{and} \quad \mathsf{x}_j \geq d'_{ij} \ . \end{aligned} \tag{21}$$

A.1.3 Class II Specific Relations

Relation (19) becomes a constraint for Class II matrices by replacing $i \neq j$ with $|i - j| = 2$.

A.1.4 Explicitly Excluding Transitions

Models for any diffusion matrix can be obtained by simply listing the relations for each forbidden transition. In order to model that a transition cannot occur, the corresponding relation is created as follows. Let $I \subseteq \{0, 1, 2, 3\}$, resp. $J \subseteq \{0, 1, 2, 3\}$ be the sets of the indices of the cells that are active in an input, resp. output column. We forbid the transition from $\bigcup_{i\in I} \mathsf{x}_i$ to $\bigcup_{j\in J} \mathsf{y}_j$ by adding the following relation

$$\Big(\sum_{i\in I} \mathsf{x}_i + \sum_{i\notin I} (1 - \mathsf{x}_i)\Big) + \Big(\sum_{j\in J} \mathsf{y}_j + \sum_{j\notin J} (1 - \mathsf{y}_j)\Big) \leq 7 \ . \tag{22}$$

A diffusion matrix should not be modeled by this approach only, as the resulting model would be quite large and probably very slow to solve. Usually, the XOR model and the branch number are modeled first, then any other non-occurring transition is forbidden.

A.1.5 Models for Class I and Class II Matrices

Various equivalent models for Class I matrices can be obtained by combining various subsets of the above relations, for instance in the original `MANTIS` and `QARMAv1` papers the matrix was defined by Relations (15), (17), (19) and (18) – in particular the XOR model is not used. We can reason as in Appendix A.1.3 to turn sets of relations into a model for Class II matrices. These models can be slightly accelerated by adding Relation (21).

A starting point consists of the XOR Relations (16) and the branch number Condition (15), removing the inadmissible transitions as described in Appendix A.1.4. For Class II matrices, we exclude eight transitions, producing a compact and fast model. However, for Class I matrices we must exclude 24 transitions, leading to a large, slow model.

For Class I matrices, the smallest model consists of Relations (15), (16), and (20). When used with Gurobi's parameters `MIPFocus` and `Cuts` both set to 2 this model results in the fastest solving times, especially for the largest problems.

A.2 Extending Solutions Inductively

For $\ell = 1$ and half-cipher, a MILP program for r is just an extension at the end of the corresponding program for $r - 1$. This ensures that the minimum active cell count in characteristics is an increasing function of r all other parameters being equal.

For $\ell = 1$ and full-cipher, it is clear that a MILP program for r is not an extension of the program for $r - 1$. This explains why sometimes the minimum active cell count in characteristics decreases for increasing r. However, the program for r is an extension of the program for $r - 2$, where the two additional rounds are added at each end of the cipher. Therefore, the minimum active cell count is a monotonic function of r restricted to the even or to the odd r. Furthermore, a MILP program for r is an extension of the corresponding program for $r - 1$ *and* φ replaced by its inverse φ^{-1} (and the roles of the two tweak blocks exchanged). So if one merges the counts for a given φ with, say, even r with the counts for φ^{-1} for odd r we obtain a monotonic sequence.

For $\ell = 2$, because of the `eXchangeRows` operations every other round, a MILP program for r is an extension of the corresponding program for $r - 2$.

We exploit these properties to provide initial starting solutions to the MILP solvers.

B Tables of Trail Weights

In Table 9, we tabulate the weights of optimal linear trails and related-tweak differential characteristics for both half-cipher and full-cipher `QARMAv2`-64, resp. `QARMAv2`-128 for various values of r, various choices of the tweak shuffle, and for $\mathscr{T} = 2$ and 1. Table 8 lists the shuffle τ_1 to τ_6, which are defined in Section 3.3 and measured in Table 9a.

Table 8: The six shuffles τ_1 to τ_6.

$$
\begin{array}{lcl}
\tau_1 & : & [\ 2,\ 10, 14,\ 6,\quad 0,\ 8,\ 12,\ 4,\quad 3,\ 11, 15,\ 7,\quad 1,\ 9,\ 13,\ 5\ ] \\
\tau_2 & : & [\ 1,\ 9,\ 13,\ 5,\quad 0,\ 8,\ 12,\ 4,\quad 2,\ 10, 14,\ 6,\quad 3,\ 11, 15,\ 7\ ] \\
\tau_3 & : & [\ 3,\ 11, 15,\ 7,\quad 2,\ 10, 14,\ 6,\quad 0,\ 8,\ 12,\ 4,\quad 1,\ 9,\ 13,\ 5\ ] \\
\tau_4 & : & [\ 2,\ 10, 14,\ 6,\quad 1,\ 9,\ 13,\ 5,\quad 0,\ 8,\ 12,\ 4,\quad 3,\ 11, 15,\ 7\ ] \\
\tau_5 & : & [\ 3,\ 11, 15,\ 7,\quad 1,\ 9,\ 13,\ 5,\quad 2,\ 10, 14,\ 6,\quad 0,\ 8,\ 12,\ 4\ ] \\
\tau_6 & : & [\ 1,\ 9,\ 13,\ 5,\quad 2,\ 10, 14,\ 6,\quad 3,\ 11, 15,\ 7,\quad 0,\ 8,\ 12,\ 4\ ]
\end{array}
$$

Table 9: Minimum number of active S-Boxes in related-tweak differential characteristics and in linear trails/non-related-value differential characteristics.

(a) Single layer cipher ($\ell = 1$, `QARMAv2`-64) with two independent tweak blocks ($\mathscr{T} = 2$).

		Half-Cipher										Full-Cipher				
	r =	3	4	5	6	7	8	9	10	11	12	2	3	4	5	6
	rounds =	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*6*	*8*	*10*	*12*	*14*
Related-tweak Differential for given φ	1	2	2	3	3	4	4	5	5	6	6	5	12	16	19	22
	τ	2	4	7	7	16	16	20	20	24	24	5	14	16	19	31
	τ^2	2	6	8	10	12	12	14	16	18	18	5	12	20	24	32
	h	2	3	7	11	14	19	25	30	34	38	5	12	23	28	41
	h_4	2	4	7	12	17	21	22	24	29	34	5	10	22	32	36
	τ_1	2	4	6	12	16	20	24	27	31	34	5	12	24	22	38
	τ_2	2	5	8	12	17	19	25	28	28	32	4	12	24	30	38
	τ_3	2	5	8	12	17	19	25	28	28	32	5	12	24	30	38
	τ_4	2	4	9	12	16	20	25	27	30	32	5	16	24	32	39
	τ_5	2	4	6	12	16	20	24	27	31	34	5	12	24	22	38
	τ_6	2	4	9	12	16	20	25	27	30	32	5	16	24	28	38
	τ_1^{-1}	2	4	6	11	15	21	24	28	30	33	5	14	16	34	32
	τ_2^{-1}	2	5	8	12	16	21	24	26	30	33	5	14	24	30	38
	τ_3^{-1}	2	5	8	12	16	21	24	26	30	33	5	14	24	30	38
	τ_4^{-1}	2	4	7	12	16	20	25	28	28	32	5	16	24	28	40
	τ_5^{-1}	2	4	6	11	15	21	24	28	30	33	5	14	16	31	32
	τ_6^{-1}	2	4	7	12	16	20	25	28	28	32	5	16	24	28	36
	τ_f	2	4	8	12	16	22	24	27	32	36	5	12	24	32	41
Linear Trails		16	23	30	35	38	41	50	57	62	67	5	32	50	64	72

(b) Two layer cipher ($\ell = 2$, `QARMAv2`-128) with two independent tweak blocks ($\mathscr{T} = 2$).

Rel.-tweak Differential	h	2	3	7	12	19	24	28	35	35	43	5	14	28	40	*51*
	h_4	2	4	9	14	18	25	27	33	37	42	5	14	28	38	*49*
	τ_f	2	4	9	14	20	24	28	32	36	40	5	12	24	40	*51*
	τ_F	2	6	11	16	26	34	44	50	*55*	*63*	5	16	32	52	*67*
Linear Trails		16	25	36	48	58	68	72	80	88	100	24	44	56	80	*104*

(c) Single layer cipher ($\ell = 1$, `QARMAv2`-64) with a single block tweak ($\mathscr{T} = 1$).

Rel.-tw. Diff.	h	5	10	14	17	24	27	31	36	39	45	8	20	31	38	47
	h_4	5	8	12	19	21	26	30	34	39	43	6	22	23	38	45
	τ_f	5	9	14	19	23	28	31	36	40	45	6	24	32	39	47

(d) Two layer cipher ($\ell = 2$, `QARMAv2`-128) with a single block tweak ($\mathscr{T} = 1$).

Rel.-tweak Differential	h	5	11	17	22	29	32	38	43	49	51	10	16	30	47	56
	h_4	5	8	18	21	29	32	39	43	49	51	6	26	31	47	53
	τ_f	5	11	14	23	27	32	37	40	47	48	6	28	32	47	48
	τ_F	5	12	20	29	41	49	59	67	–	–	6	26	44	67	–
	rounds =	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*6*	*8*	*10*	*12*	*14*
	r =	3	4	5	6	7	8	9	10	11	12	2	3	4	5	6
		Half-Cipher										Full-Cipher				

C A Remark on Minimal Weight Linear Trails

In Figure 10, we display a low weight iterative cell-based linear trail for single-layer `QARMAv2`. It involves 34 active cells across six rounds. On a full trail, the total weight could be reduced by up to eight cells if the first or last round have a weight of nine, allowing at each end of the trail the replacement of up to two weight nine rounds with weight seven rounds.

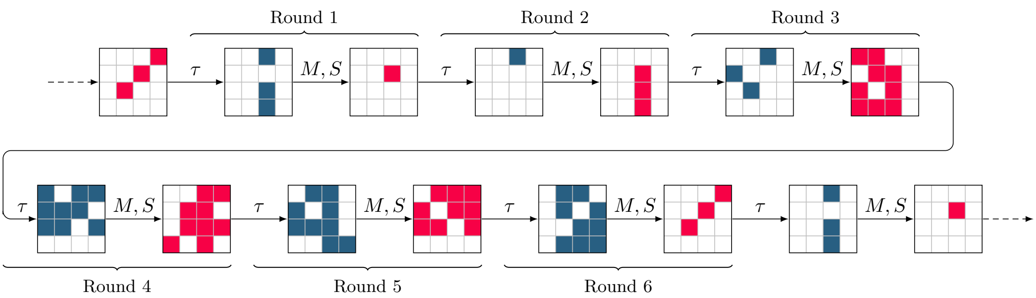


Figure 10: Example of minimal weight iterative linear trail.

This results in an asymptotic average of around 5.67 cells per round. In contrast, an MDS matrix-based approach has a minimum of 25 cells over four rounds, averaging 6.25 cells per round. To achieve a similar minimal active cell count, using a Class I Almost MDS matrix and the `MIDORI` `StateShuffle` would need about 10% more rounds than an MDS matrix with an `AES`-like `ShiftRows`. However, the former diffusion matrix is lighter, potentially leading to a more compact and faster design.

D Characteristics for Differential, Impossible Differential, and Boomerang Attacks

We provide detailed characteristics for selected cryptanalytic results of Section 4: Figure 11 illustrates how a key-recovery attack on $r = 6$ rounds of `QARMAv2-64` could be designed based on a characteristics following the optimal truncated differential pattern for $r = 4$. Figure 12 illustrates the longest impossible differential distinguisher for `QARMAv2-64` under a cell-based miss-in-the-middle approach. Figures 13 to 15 show some potential boomerang distinguishers identified by a cell-wise model for a total of 12 rounds. The more balanced setups yield lower bounds.

E Full Diffusion Property for the Two-Layer Version

In the following (as well as in Appendix F), it is understood that "perturbed" means "influenced non-linearly" (by a fixed input bit). A cell is perturbed if at least one of its output bits is perturbed.

The four-round full diffusion property for the cipher with $\ell = 2$ can be easily proved with the aid of a simple computer program. The program starts with just one perturbed state cell, and keeps track of which cells are perturbed after each round operation. We display in Figure 16 the trails corresponding to the four cells in the first row of the state in the case where `eXchangeRows` occurs after the second full round, and in the case where `eXchangeRows` occurs after the first full round. These two cases correspond to the initial structure of the cipher for even and odd r, respectively. Note that `eXchangeRows` and the S-Box layer commute. The trails which have been omitted are very similar to the displayed ones, and all end with 32 perturbed cells.

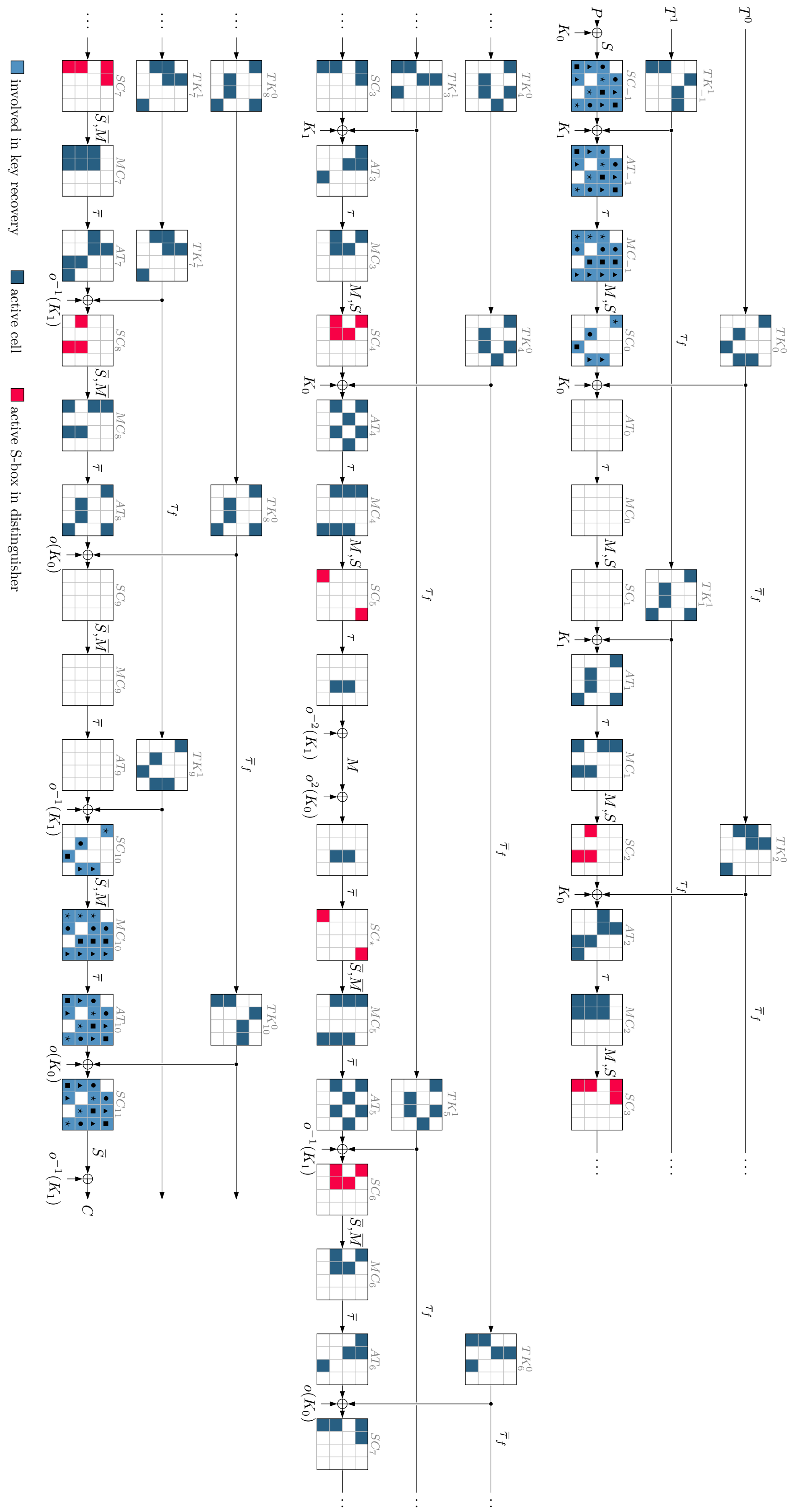


Figure 11: Sketch of key recovery attack on QARMAv2 with $r = 6$ based on a differential distinguisher for $r = 4$.

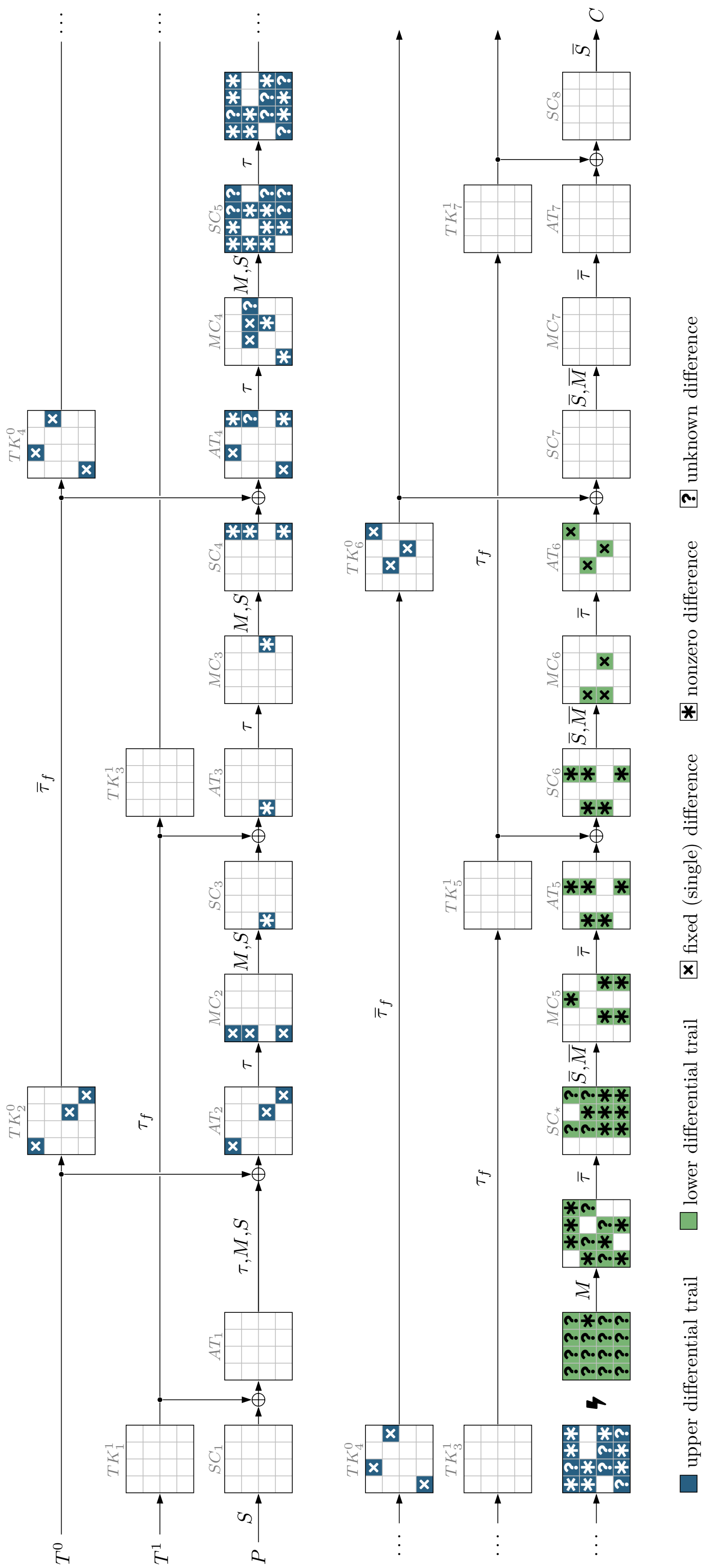


Figure 12: Longest impossible differential distinguisher for QARMAv2.

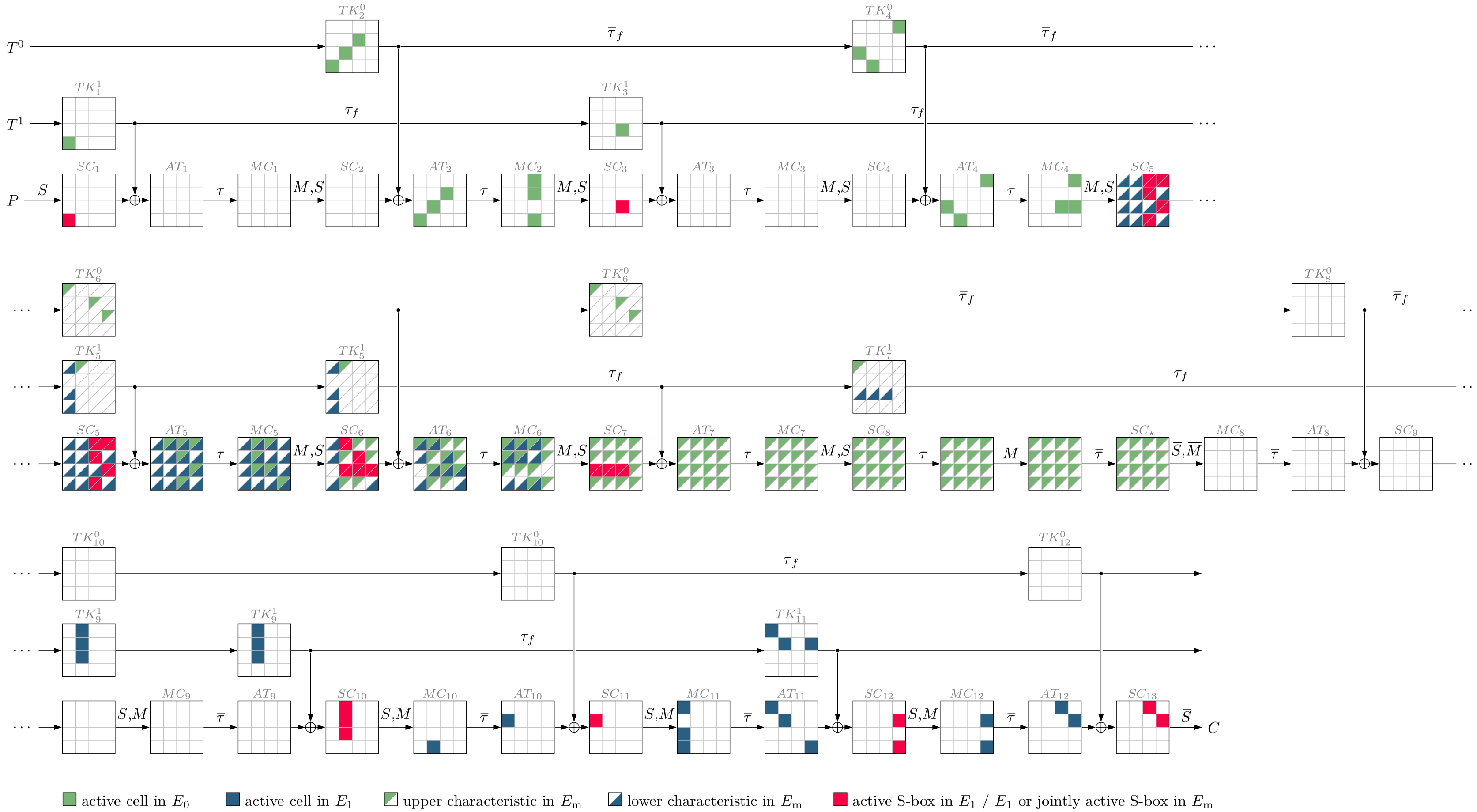


Figure 13: Boomerang distinguisher of weight 73 for $7 + 5$ rounds of QARMAv2-64, corresponding to a probability bound of $2^{-10\times 4} \cdot (\frac{10}{16})^{13} \approx 2^{-48.8}$.

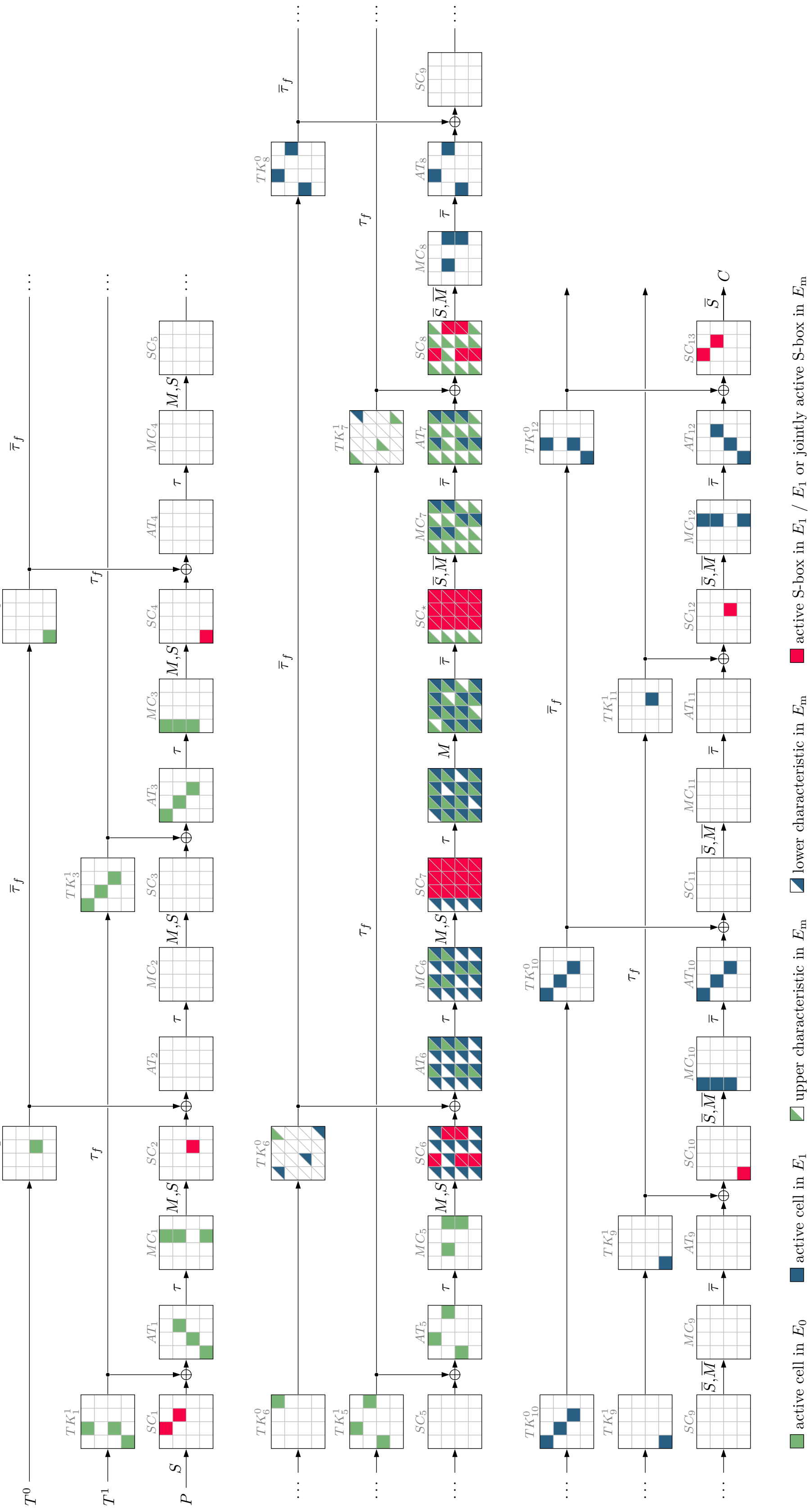


Figure 14: Boomerang distinguisher of weight 82 for 6 + 6 rounds of QARMAv2-64, corresponding to a probability bound of $2^{-8\times 4} \cdot (\frac{10}{16})^{34} \approx 2^{-55.1}$.

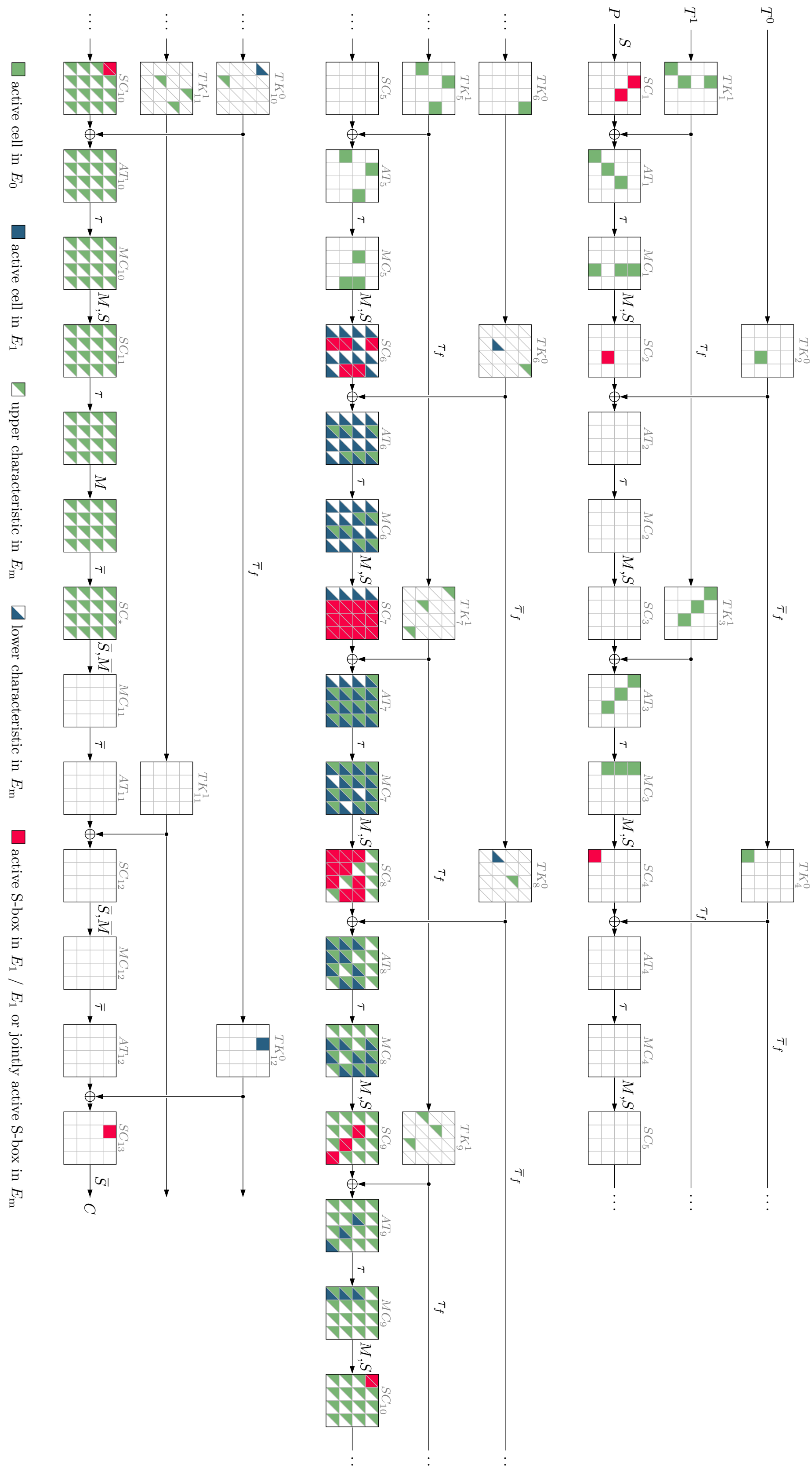


Figure 15: Boomerang distinguisher weight 60 for $10+2$ rounds of QARMAv2-64, corresponding to a probability bound of $2^{-5\times 4} \cdot (\frac{10}{16})^{30} \approx 2^{-40.3}$.

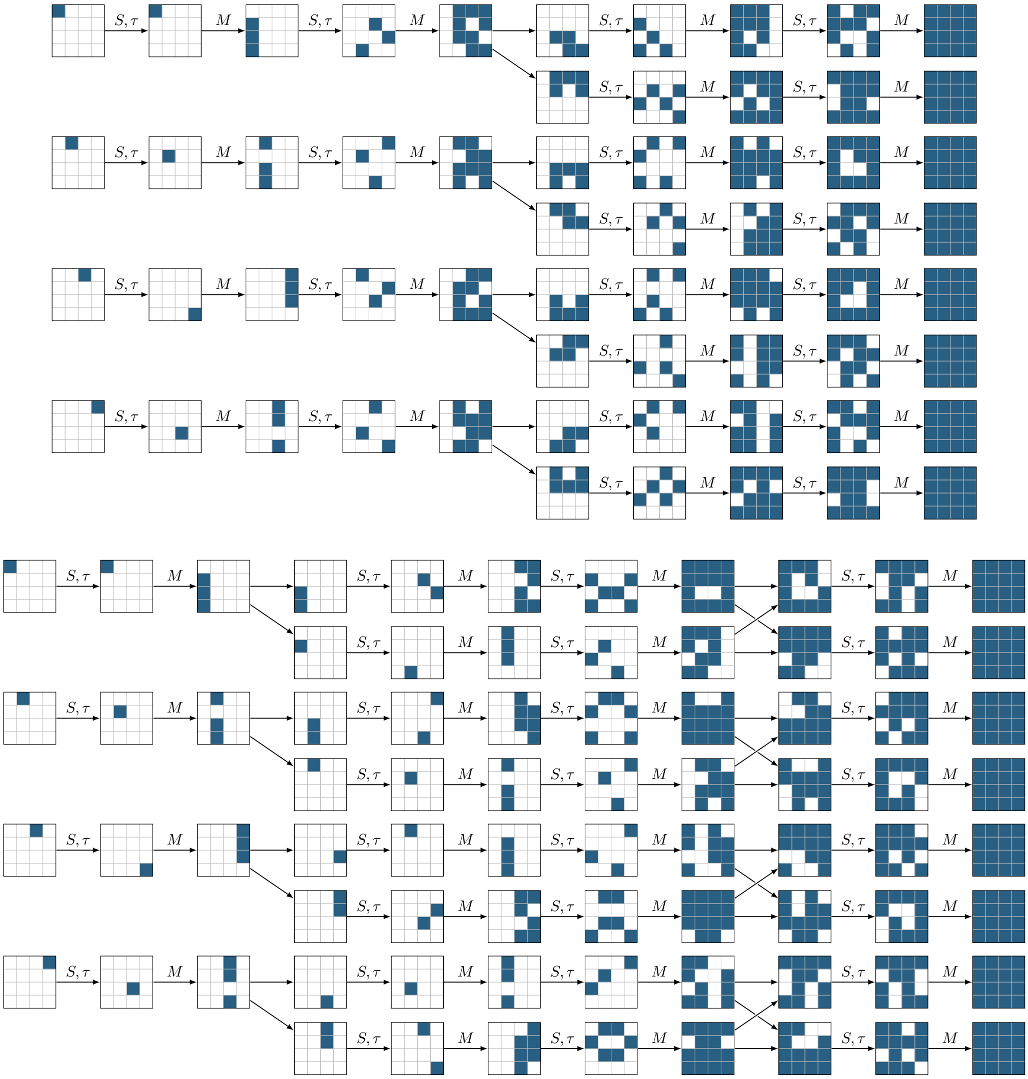


Figure 16: Selected trails from the proof of the four-round full diffusion property for $\ell = 2$.

In order to verify whether all bits of the state are perturbed after four rounds, we now count how many bits in each cell are perturbed after each S-Box layer.

Recall from [Ava17] that σ_0 satisfies a weaker form of Property (iv) in Section 3.4, namely: *There is one input bit that perturbs only three output bits, and all other three input bits perturb all four output bits.* It is now easy to prove that full diffusion after four rounds holds with σ_0: After the first, resp. second, S-Box layer, at least three, resp. four bits of each perturbed cell are individually perturbed. With ק, all four bits in each perturbed cell are perturbed already after the first S-Box layer.

The property holds also for the 8-bit S-Box construction from `QARMAv1` [Ava17]: after the first, second and third S-Box layer, the number of perturbed bits in the perturbed cells is at least three, seven, and eight. In Appendix F a 16-bit S-Box is discussed.

F Wide Block Versions

In the introduction, we justified the choice of designing a TBC with short blocks of just 64 and 128 bits, but with a long "native" tweak to express a larger permutation space. However, the ideas in this paper are more general, and we showcase their flexibility by sketching 256-bit and 512-bit block versions of `QARMAv2`. These are just concepts and should not be taken for mature proposals – for instance, a full cryptanalysis is missing. However, we believe that discussing them is timely, because of the NIST's recent interest in tweakable wide block encryption techniques [NIS23].

The two examples have two layers, and use 8-bit and 16-bit S-Boxes in place of the 4-bit S-Boxes used by `QARMAv2`-64 and `QARMAv2`-128. The 8-bit S-Box follows `QARMAv1`'s design, cf. Figure 17. The 16-bit S-Box similarly consists of four parallel S-Boxes with interwoven outputs, cf. Figure 18. We can assume that keys and tweaks are single blocks, and everything else is easily adapted from the main design. We expected that these variants will have over `AESQ` and `Pholkos` roughly the same advantage (in the sense of Section 6) that `QARMAv2`-128 enjoys over the `AES`.

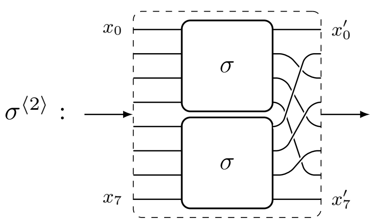


Figure 17: The eight-bit S-Box.

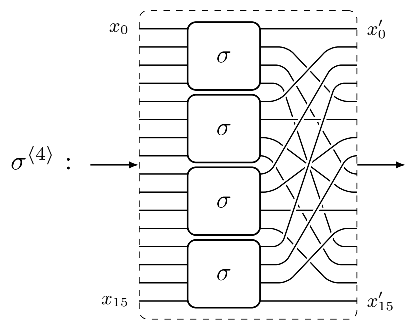


Figure 18: The sixteen-bit S-Box.

For the 16-bit S-Box $\sigma_0^{\langle 4\rangle}$ (four copies of σ_0), in Table 10 we show the minimal counts of the perturbed bits in each cell, up to reordering of the four 4-bit subcells. Therefore, with $\sigma_0^{\langle 4\rangle}$, the four-round full diffusion property for $\ell = 2$ holds, but the three-round full diffusion property for $\ell = 1$ does not (a fourth round is needed). With $\mathsf{P}^{\langle 4\rangle}$, all 16 bits are perturbed already after the second S-Box layer, whence Theorem 2 holds in full generality.

Table 10: Lower bounds for the number of perturbed bits per perturbed cell in the composite 16-bit S-Box $\sigma_0^{\langle 4\rangle}$, after each S-Box layer, with the values after the 4-bit S-Boxes and the bit interweaving ("Weave") given separately.

	Start	First Layer		Second Layer				Third Layer		Fourth Layer
		S-Box	Weave	S-Box	⌐ Weave		¬	S-Box	Weave	S-Box
Counts for 4-Bit Cells	1	3	1	3	3	3	3	4	3	4
	0	0	1	3	3	3	2	4	3	4
	0	0	1	3	3 ,	2 or	2	4	3	4
	0	0	0	0	0	1	2	0	3	4
Total	1	3		9				12		16

Regarding the diffusion matrix, it can be proved that $\mathrm{circ}(0, \rho^4, \rho^8, \rho^{12})$ on $(\mathbb{F}_2^{16})^4$ is a Class I matrix (see Section 2.5 and Appendix H).

G Cryptanalysis of QARMAv1

For the reader's convenience, we collect the published cryptanalysis of QARMAv1 in Table 11, including time, memory, and data information. Some of the attacks break the security claims for QARMAv1 on reduced round versions of the cipher, but the residual security margin is quite ample, at least 4, resp. 12 rounds for QARMAv1-64_7, resp. QARMAv1-128.

Table 11: Cryptanalysis of QARMAv1. The number of rounds is given as $x + y$, where x, resp. y is the number of S-Box layers before, resp. after the reflector. An asterisk near the size means that $M_{4,1}$ is the diffusion matrix, not $M_{4,2}$.

Size	Rounds attacked	Outer whitening	Attack Complexity Time	Data	Memory	Technique	Reference
64	$4+6$	N	$2^{116}+2^{70.1}$	2^{53} CP	2^{116}	MITM	[ZD16]
64	$3+8$	N	2^{69}	$2^{58.38}$ CP	$2^{63.38}$	Imp. Diff.	[LZG+20]
64	$4+4$	Y	$2^{33}+2^{90}$	2^{16} CP	2^{90}	MITM	[LJ18]
64	$4+5$	Y	$2^{48}+2^{89}$	2^{16} CP	2^{89}	MITM	[LJ18]
64	$4+6$	Y	2^{72}	2^{61} CP	$2^{78.2}$ bits	Trunc. Imp. Diff.	[YQC18]
64	$4+6$	Y	2^{59}	2^{59} KP	$2^{29.6}$ bits	Rel-Tweak Stat. Sat.	[LHW19]
64	$4+6$	Y	$2^{75.13}$	$2^{47.12}$ CP	2^{72}	Rel-Tweak Trunc. Diff.	[SII23]
64	$5+5$	Y	Time $\times$ Data (CP) $= 2^{125.8}$		2^{37} bits	Rel-Tweak Imp. Diff.	[ZD19]
64	$5+5$	Y	$2^{83.53}$	$2^{47.06}$ CP	2^{80}	Rel-Tweak Trunc. Diff.	[SII23]
64	$3+8$	Y	$2^{64.4}+2^{80}$	2^{61} CP	2^{61}	Imp. Diff.	[ZDW18]
64	$4+7$	Y	$2^{120.4}$	2^{61} CP	2^{116}	Trunc. Imp. Diff.	[YQC18]
64	$5+6$	Y	$2^{111.16}$	$2^{34.26}$ CP	2^{108}	Rel-Tweak Trunc. Diff.	[SII23]
64	$4+8$	Y	$2^{66.2}$	$2^{48.4}$ CP	$2^{53.70}$	Zero Corr./Integral	[ADG+19]
128	$4+6$	N	$2^{232}+2^{141.7}$	2^{105} CP	2^{232}	MITM	[ZD16]
128	$3+8$	N	2^{137}	$2^{111.38}$ CP	$2^{120.38}$	Imp. Diff.	[LZG+20]
128	$3+8$	N	$2^{145.98}$	$2^{102.54}$ CP	$2^{135.54}$	Rel-Tweak Imp. Diff.	[DWLW22]
128*	$4+6$	Y	$2^{237.3}$	2^{122} CP	2^{144}	Trunc. Imp. Diff.	[YQC18]
128*	$4+7$	Y	$2^{241.8}$	2^{122} CP	2^{232}	Trunc. Imp. Diff.	[YQC18]
128	$2+8$	Y	$2^{120.94}$	$2^{104.02}$ CP	$2^{94.50}$	Rel-Tweak Imp. Diff .	[DWLW22]
128	$5+5$	Y	2^{156}	2^{88} CP	2^{152} bits	MITM	[LJ18]
128	$5+5$	Y	$2^{164.48}$	2^{88} CP	2^{97}	MITM	[LZG+20]
128	$4+7$	Y	$2^{126.1}$	$2^{126.1}$ KP	2^{71} bits	Rel-Tweak Stat. Sat.	[LHW19]
128	$4+8$	Y	$2^{156.06}$	2^{88} CP	2^{154}	MITM	[LZG+20]
128	$8+3$	Y	$2^{104.60}$	$2^{124.05}$ CP	2^{48}	Rel-Tweak Trunc. Diff.	[SII23]
128	$8+4$	Y	$2^{154.53}$	$2^{108.52}$ CP	2^{144}	Rel-Tweak Trunc. Diff.	[SII23]
128	$9+4$	Y	$2^{238.02}$	$2^{106.63}$ CP	2^{240}	Rel-Tweak Trunc. Diff.	[SII23]

In light of the changes made to QARMAv2, we expect that the same attacks on the new design will have higher complexities.

H Properties of the Diffusion Matrices

This Appendix is an abridged version of Section 3.1 of [Ava17], with some additional remarks. For the reader's convenience, we report without proofs the theorems on the classification of the Almost-MDS matrices used in QARMAv1 and QARMAv2, and their properties.

The complexity of QARMAv1's and QARMAv2's diffusion layers comes from the matrix. Roughly speaking, the usual requirements on a diffusion matrix are:

1. It should guarantee mathematically provable good diffusion; and
2. It should be as lightweight as possible.

The first requirement is usually quantified by the *branch number* [DR02]. MDS matrices, which are those with maximal branch number, i.e. equal to their dimension plus one, have been used for decades in symmetric primitive design [SV94, Vau94]. However, recent designs have focused on lighter matrices, such as the *Almost MDS* ones, with branch number equal to their dimension. Using such matrices, achieving similar security as designs using MDS matrices may require more rounds, but, with a suitable choice of components, the Almost MDS-based design *can* also be smaller and faster.

The second requirement is often seen as *minimizing the matrix weight.* This reading of the requirement is aimed at software implementations and immediately leads, for instance, to the search for matrices with many ones [JV04]. However, for latency-critical hardware implementations, we want to obtain the *smallest circuit depth* instead. This is equivalent to *minimizing the maximum of the weights of the rows* of the binary matrix describing the action of the original r by r matrix as a linear operator over $\mathbb{F}_2^{mr}$, m being the S-Box size.

Matrices whose entries are all either 0 or 1, i.e. $\{0,1\}$-matrices, are clearly optimal according to the second requirement. Since a MDS matrix cannot have vanishing entries, a MDS $\{0,1\}$-matrix would have to be an all-ones matrix, which is absurd. Hence $\{0,1\}$-matrices cannot be MDS. On the other hand, Almost MDS examples can be found, but in dimensions 2, 3, and 4 only [CK08]. Examples are given by matrices that have zeros on the main diagonal and ones everywhere else. However, they must be used with care: In `MIDORI`-64, they enable the propagation of iterative characteristics rooted in the fixed-point structure of the S-Box by creating parallel copies of the same characteristics. We must look for diffusion layers that prevent this phenomenon.

If the S-Box outputs undergo suitable distinct linear transformations for each target cell prior to begin added together, the resulting characteristics will differ. This differentiation reduces the likelihood of both characteristics propagating through the subsequent S-Box layer. Consequently, we explore options beyond $\{0,1\}$-matrices. This is usually entails considering matrices over binary extension fields.

A problem with matrices over binary extension fields is that multiplication by any element different from 0 and 1 needs a modular reduction step, which adds latency. However, we can consider the quotient ring $R_m = \mathbb{F}_2[X]/(X^m + 1)$. The image ρ of X in the ring R_m satisfies $\rho^m = 1$, and $\{\rho^{m-1}, \ldots, \rho^2, \rho, 1\}$ is a basis for R_m as a $\mathbb{F}_2$-algebra. The multiplication by ρ in the ring R_m with respect to this basis is just a simple circular left rotation of the coordinates. Since these coordinates are elements of $\mathbb{F}_2$, i.e. bits, an action of ρ on m-bit cells is immediately defined. Since R_m contains zero divisors (for $m \geq 2$), care shall be taken when constructing invertible matrices. However, if the search is restricted to the circulants of the form

$$M = \operatorname{circ}(0, \rho^a, \rho^b, \rho^c) = \begin{pmatrix} 0 & \rho^a & \rho^b & \rho^c \\ \rho^c & 0 & \rho^a & \rho^b \\ \rho^b & \rho^c & 0 & \rho^a \\ \rho^a & \rho^b & \rho^c & 0 \end{pmatrix} , \tag{23}$$

then Almost MDS matrices are not exceedingly difficult to find, as shown by the following Theorem.

Theorem 3. *Let $R_m = \mathbb{F}_2[\rho]$ be the quotient ring $\mathbb{F}_2[X]/(X^m + 1)$ where ρ is the image of X in R, $m \geq 2$. The Almost MDS matrices M of the form (23) over the ring R_m are precisely the invertible ones, i.e. are those for which $\rho^{4a} + \rho^{4b} + \rho^{4c} \in R_m^*$.*

Remark 6. If $m = 4$ or 8, then *all* matrices (23) are invertible. In fact, for $m = 4$ the determinant $\rho^{4a} + \rho^{4b} + \rho^{4c}$ is always 1 and for $m = 8$ it is equal to either 1 or ρ^4.

Now let us consider the matrices of type (23) with equally lightweight inverse.

Theorem 4. *Let $R_m = \mathbb{F}_2[X]/(X^m + 1) = \mathbb{F}_2[\rho]$ be defined as in Theorem 3. The Almost MDS matrices $M = \operatorname{circ}(0, \rho^a, \rho^b, \rho^c)$ that admit an inverse of the same form $\overline{M} = \operatorname{circ}(0, \rho^d, \rho^e, \rho^f)$, i.e. with entries of weight at most one, are those that satisfy*

$a \equiv c + t$ where $2\,t \equiv 0 \bmod m$ (for odd m this implies $t = 0$, for even m it can be $t = 0$ or $m/2$). In this case, the parameters of the matrix $\overline{M}$ are: $d \equiv a - 2\,b$, $e \equiv -b$, and $f \equiv d + t \bmod m$.

Such a matrix M is involutory if and only if $2\,b \equiv 0$.

Remark 7. An analogue of Theorem 3 holds over fields $\mathbb{F}_{2^m}$ as well. A matrix $M = \mathrm{circ}(0, A, B, C)$ with $ABC \neq 0$ is invertible if and only if its determinant $A^4+B^4+C^4 \neq 0$, in which case it is Almost MDS. However, the multiplications by A, B, or C are circuits of non-negligible depth, unless they are equal to 1.

Besides the `MIDORI` circulant $M_0 := \mathrm{circ}(0, 1, 1, 1)$ several other matrices have been considered. For `QARMAv1`-64, and both versions of `QARMAv2`, these include

$$\begin{aligned} M_{4,1} &= Q_{4,1} = \mathrm{circ}(0, \rho, \rho^2, \rho^3) \ , \\ M_{4,2} &= Q_{4,2} = \mathrm{circ}(0, \rho, \rho^2, \rho) \ , \text{ and} \\ M_{4,3} &= Q_{4,3} = \mathrm{circ}(0, 1, \rho^2, 1) \ . \end{aligned}$$

For `QARMAv1`-128 the considered matrices were

$$\begin{aligned} M_{8,1} &= Q_{8,1} = \mathrm{circ}(0, \rho^2, \rho^4, \rho^6) \ , \\ M_{8,2} &= Q_{8,2} = \mathrm{circ}(0, \rho, \rho^4, \rho^5) \ , \\ M_{8,3} &= Q_{8,3} = \mathrm{circ}(0, 1, \rho^4, 1) \ , \text{ and} \\ M_{8,4} &= \mathrm{circ}(0, \rho, \rho^2, \rho^5) \text{ with } Q_{8,4} = \mathrm{circ}(0, \rho, \rho^4, \rho) \ . \end{aligned}$$

Note that $M_{8,1}$, $M_{8,2}$, $M_{8,3}$, and $Q_{8,4}$ are involutory, whereas $\overline{M}_{8,4} = \mathrm{circ}(0, \rho^5, \rho^6, \rho) = \rho^4 \cdot M_{8,4}$. The matrices M_* and their inverses are used in the forward and backward functions, and the Q_*'s are used in the reflector. These matrices fall into two classes:

1. *Class I* includes M_0, $M_{4,1}$ and $M_{8,1}$; and
2. *Class II* includes $M_{4,2}$, $M_{4,3}$, $M_{8,2}$, $M_{8,3}$, $M_{8,4}$ and $Q_{8,4}$.

Class I matrices have 51 possible active column-to-column state transitions, of which the 67 transitions of Class II matrices are a superset. These transitions are displayed in Figure 9. Exhaustive enumeration yields the following result.

Proposition 1. *All the matrices over R_4 and R_8 characterised in Theorem 4 (i.e. the circulants $M = \mathrm{circ}(0, \rho^a, \rho^b, \rho^c)$ with an inverse of the same form) admit only the column-to-column transitions shown in Figure 9.*

Remark 8. As noted in Section 2.5, Class I matrices offer improved diffusion over Class II matrices: This advantage applies to both linear trails and related-tweak differential characteristics using both `QARMAv1`'s and `QARMAv2`'s tweak schedules. In `QARMAv1`, the selection of the Class II matrix shifted from $M_{4,1}$ to $M_{4,2}$ in early design, despite the higher active cell counts of the former matrices. This was done because the four selection criteria were evaluated together:

(i) The number of fixed points of the matrix used in the reflector;

(ii) The number of active S-Boxes in linear and related-tweak differential trails, which depends on the matrix class;

(iii) The differential uniformity of the construction consisting of an S-Box layer, the diffusion matrix, and a second S-Box layer, where only two input cells are active; and

(iv) The applicability of the analysis of invariant subspace attacks in [Ava17, Section 4.3].

For `QARMAv1`, a much better result in the third criterion motivated the choice of $M_{4,2}$, the relative improvement being larger than in the first two criteria: With hindsight, these heuristics are explained by Tim Beyne's observation in [Bey23]. For `QARMAv2`, the choice of Class I matrix $M_{4,1}$ was backed by a more comprehensive cryptanalysis, and the study performed for `QARMAv2` supersedes Criterion (i) and improves on Criteria (ii) to (iv). This analysis in turn required the selection a different S-Box compared to `QARMAv1` in order to reap the benefits of improved active cell counts without losing in terms of potentially unbound subspace trails.

I Test Vectors

I.1 `QARMAv2`-64

`P = 0000000000000000`

`K0,K1 = 0123456789abcdef,fedcba9876543210`
`T0,T1 = 7e5c3a18f6d4b290,1eb852fc9630da74`

For $r = 4$: `C = 2cc660354929f2ca`
For $r = 5$: `C = 76d5422b082e32ad`
For $r = 6$: `C = ca1ad3689c2b8e8d`
For $r = 7$: `C = 24b73800025aa50f`
For $r = 8$: `C = a3990d50099ef616`
For $r = 9$: `C = d459510ab82c66fc`

I.2 `QARMAv2`-128

`P = 00000000000000000000000000000000`

`K0,K1 = 00102030405060708090a0b0c0d0e0f0,0f0e0d0c0b0a09080706050403020100`
`T0,T1 = 7e5c3a18f6d4b290e5c3a18f6d4b2907,1eb852fc630da741b852fc960da741eb`

For $r = 9$: `C = 361262e2ecf88f03f4ea898d6a4f412f`
For $r = 11$: `C = a874291a195606e17805e5bd05f8d066`
For $r = 13$: `C = 5f514df8ac6cd2c51c23b8e62e6a2d6a`
For $r = 15$: `C = 8c3471d4ebb28b6ecb4ed28586c77f8e`

I.3 `QARMAv2`-64-σ_0

`P = 0000000000000000`

`K0,K1 = 0123456789abcdef,fedcba9876543210`
`T0,T1 = 7e5c3a18f6d4b290,1eb852fc9630da74`

For $r = 4$: `C = 88f2efe5b4e8a4fc`
For $r = 5$: `C = 5eba5ebd2ebc06e3`
For $r = 6$: `C = ad6ca9c3f2bdd37e`