QARMAv2

Roberto Avanzi, Subhadeep Banik, Orr Dunkelman, Maria Eichlseder
Shibam Ghosh, Marcel Nageler, and Francesco Regazzoni
Arm, CRI, Universities of Amsterdam, Graz, Haifa, Lugano

arm

Introduction

arm

What is QARMAv2?

What is QARMAv2?

QARMAv2 is a revision of the Tweakable Block Cipher QARMAv1 from FSE 2017 to improve its security and allow for longer tweaks, while keeping latency and area similar.

Like QARMAv1, it is in the public domain, no IPR exerted on any component of it by any party that worked on the design!

☞ http://eprint.iacr.org/2023/929 ☜

arm

Why QARMAv2?

I mean, QARMAv1 looks fine, so why update it?

Cipher	Rounds Attacked	Outer Whitening?	Attack Complexity Time	Data	Memory	Technique	Ref.
64	4 + 6	N	$2^{116} + 2^{70.1}$	2^{53} CP	2^{116}	MITM	[ZD16]
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 + 7	Y	$2^{120.4}$	2^{61} CP	2^{116}	trunc. imp. diff.	[YQC18]
64	3 + 8	Y	$2^{64.4} + 2^{80}$	2^{61} CP	2^{61}	imp. diff.	[ZDW18]
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	5 + 5	Y	2^{156}	2^{88} CP	2^{152} bits	MITM	[LJ18]
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	4 + 7	Y	$2^{126.1}$	$2^{126.1}$ KP	2^{71} bits	rel-tweak stat. sat.	[LHW19]

Why QARMAv2?

- Not a whim or just to `papers++`:

  Since the introduction of QARMAv1, after many years of research but also trial and error, we achieved a better understanding of how to design block ciphers, and of the requirements coming from practical applications.

- Longer tweaks for applications, flexibility, and security.

- New choice of some components to improve security.

arm

Why QARMAv2?

- Not a whim or just to `papers++`:

  Since the introduction of QARMAv1, after many years of research but also trial and error, we achieved a better understanding of how to design block ciphers, and of the requirements coming from practical applications.

- Longer tweaks for applications, flexibility, and security.

- New choice of some components to improve security.

In a nutshell: 1) More flexible inputs...

- QARMAv2-64-128: 64-bit block size and 128 bit key, and tweaks up to 128 bits (up from 64 bits)
- QARMAv2-128-*s*: 128-bit block size and *s* bit key, with s = 128, 192 or 256, and tweaks up to 256 bits (up from 128 bits)
- QARMAv2-64-128 for Pointer and Memory Authentication (uses a lighter S-Box)

In a nutshell: 2) Security bounds...

To align with common requirements from NIST and other SDOs we want to move from the tradeoff definition of security

$$\text{Time} \times \text{Data} \geq 2^{128-\varepsilon} \text{ or } 2^{256-\varepsilon}$$

of `PRINCE`, `MANTIS`, `QARMAv1`, etc... to

$$\text{if Data} \leq 2^{56 \text{ resp. } 80}, \text{ then Time} \geq 2^{128 \text{ resp. } 128,\ 192, \text{ or } 256}$$

similarly to `PRINCEv2`.
Achieving this requires changes in the structure.

arm

In a nutshell: 2) Security bounds...

To align with common requirements from NIST and other SDOs we want to move from the tradeoff definition of security

$$\text{Time} \times \text{Data} \geq 2^{128-\varepsilon} \text{ or } 2^{256-\varepsilon}$$

of `PRINCE`, `MANTIS`, `QARMAv1`, etc... to

$$\text{if Data} \leq 2^{56 \text{ resp. } 80}, \text{ then Time} \geq 2^{128 \text{ resp. } 128,\ 192, \text{ or } 256}$$

similarly to `PRINCEv2`.

Achieving this requires changes in the structure.

In a nutshell: 2) Security bounds...

To align with common requirements from NIST and other SDOs we want to move from the tradeoff definition of security

$$\text{Time} \times \text{Data} \geq 2^{128-\varepsilon} \text{ or } 2^{256-\varepsilon}$$

of `PRINCE`, `MANTIS`, `QARMAv1`, etc... to

$$\boxed{\text{if Data} \leq 2^{56 \text{ resp. } 80}, \text{ then Time} \geq 2^{128 \text{ resp. } 128,\ 192, \text{ or } 256}}$$

similarly to `PRINCEv2`.
Achieving this requires changes in the structure.

Security Considerations

Security – Stateless IVs, Modes, Memory Encryption

- AES with a 128-bit block in a XEX construction and a 128-bit block, 128-bit tweak TBC like QARMAv1 have something in common.

  Syntetic or random IVs do not work well: Collision after $O(2^{64})$ messages.

  Worse with modes like GCM, with a 96-bit IV and a 32-bit counter.

- One solution is to use longer blocks.

  However, a 256-bit wide cipher can be heavier than a 128-bit cipher.

  Potentially slower full diffusion. So, maybe even more rounds overall.

arm

Security – Stateless IVs, Modes, Memory Encryption

- AES with a 128-bit block in a XEX construction and a 128-bit block, 128-bit tweak TBC like QARMAv1 have something in common.

  Syntetic or random IVs do not work well: Collision after $O(2^{64})$ messages.

  Worse with modes like GCM, with a 96-bit IV and a 32-bit counter.
- One solution is to use longer blocks.

  However, a 256-bit wide cipher can be heavier than a 128-bit cipher.

  Potentially slower full diffusion. So, maybe even more rounds overall.

Security – Stateless IVs, Modes, Memory Encryption

- AES with a 128-bit block in a XEX construction and a 128-bit block, 128-bit tweak TBC like QARMAv1 have something in common.

  Syntetic or random IVs do not work well: Collision after $O(2^{64})$ messages.

  Worse with modes like GCM, with a 96-bit IV and a 32-bit counter.
- ~~One solution is to use longer blocks.~~
- Remark: a 128-bit block cipher with 256-bit tweaks allows to define a space of 2^{256} permutations for each value of the key.

  So, for Cryptographic Memory Encryption, we can have 64-bit counters, 64-bit addresses, 64 bits of "realm identity," and room to spare.
- Not to speak of 128-bit address spaces. Hence 256-bit tweaks are desirable.
- For embedded: 64-bit blocks, and 128-bit keys and tweaks should be ok.

Security – Stateless IVs, Modes, Memory Encryption

- AES with a 128-bit block in a XEX construction and a 128-bit block, 128-bit tweak TBC like QARMAv1 have something in common.

  Syntetic or random IVs do not work well: Collision after $O(2^{64})$ messages.

  Worse with modes like GCM, with a 96-bit IV and a 32-bit counter.
- ~~One solution is to use longer blocks.~~
- Remark: a 128-bit block cipher with 256-bit tweaks allows to define a space of 2^{256} permutations for each value of the key.

  So, for Cryptographic Memory Encryption, we can have 64-bit counters, 64-bit addresses, 64 bits of “realm identity,” and room to spare.
- Not to speak of 128-bit address spaces. Hence 256-bit tweaks are desirable.
- For embedded: 64-bit blocks, and 128-bit keys and tweaks should be ok.

arm

Security – Stateless IVs, Modes, Memory Encryption

- AES with a 128-bit block in a XEX construction and a 128-bit block, 128-bit tweak TBC like QARMAv1 have something in common.

  Syntetic or random IVs do not work well: Collision after $O(2^{64})$ messages.

  Worse with modes like GCM, with a 96-bit IV and a 32-bit counter.
- ~~One solution is to use longer blocks.~~
- Remark: a 128-bit block cipher with 256-bit tweaks allows to define a space of 2^{256} permutations for each value of the key.

  So, for Cryptographic Memory Encryption, we can have 64-bit counters, 64-bit addresses, 64 bits of "realm identity," and room to spare.
- Not to speak of 128-bit address spaces. Hence 256-bit tweaks are desirable.
- For embedded: 64-bit blocks, and 128-bit keys and tweaks should be ok.

Separate key and tweak schedules (as in QARMAv1)

- TWEAKEY? But it unifies key and tweak in a single, undifferentiated input. May not reflect the different real-world security requirements on keys, tweaks.
- With a TBC, the key is changed infrequently. Its generation can thus be hardened without impact on overall performance. Hence, we may not need to consider related-key attacks.
- Conversely, the tweak changes often, it is public and the adversary may even be capable to choose its value. This forces consideration of related-tweak attacks.
- With a unified schedule, the cryptanalysis may overestimate the number of rounds required to reach a target security level.
- These considerations prompt us to keep separate key and tweak schedules.

Separate key and tweak schedules (as in QARMAv1)

- TWEAKEY? But it unifies key and tweak in a single, undifferentiated input.
  May not reflect the different real-world security requirements on keys, tweaks.
- With a TBC, the key is changed infrequently.
  Its generation can thus be hardened without impact on overall performance.
  Hence, we may not need to consider related-key attacks.
- Conversely, the tweak changes often, it is public and the adversary may even be capable to choose its value. This forces consideration of related-tweak attacks.
- With a unified schedule, the cryptanalysis may overestimate the number of rounds required to reach a target security level.
- These considerations prompt us to keep separate key and tweak schedules.

Separate key and tweak schedules (as in QARMAv1)

- TWEAKEY? But it unifies key and tweak in a single, undifferentiated input.
  May not reflect the different real-world security requirements on keys, tweaks.
- With a TBC, the key is changed infrequently.
  Its generation can thus be hardened without impact on overall performance.
  Hence, we may not need to consider related-key attacks.
- Conversely, the tweak changes often, it is public and the adversary may even be capable to choose its value. This forces consideration of related-tweak attacks.
- With a unified schedule, the cryptanalysis may overestimate the number of rounds required to reach a target security level.
- These considerations prompt us to keep separate key and tweak schedules.

Separate key and tweak schedules (as in QARMAv1)

- TWEAKEY? But it unifies key and tweak in a single, undifferentiated input.
  May not reflect the different real-world security requirements on keys, tweaks.
- With a TBC, the key is changed infrequently.
  Its generation can thus be hardened without impact on overall performance.
  Hence, we may not need to consider related-key attacks.
- Conversely, the tweak changes often, it is public and the adversary may even be capable to choose its value. This forces consideration of related-tweak attacks.
- With a unified schedule, the cryptanalysis may overestimate the number of rounds required to reach a target security level.
- These considerations prompt us to keep separate key and tweak schedules.

Security – Better Key and Tweak Schedules

We move from Even-Mansour to an Alternating-Key Schedule because:

- Security bounds are better and more “normal” (as already seen).
- Longer tweak ⇒ the adversary has more control on the internal states.
- Hence, we may need more rounds if we kept the Even-Mansour scheme.
- A better key schedule may help balance things.

In fact, we shall also see that a better tweak schedule can allow longer tweaks at no extra cost in terms of rounds needed (at least in some cases).

Security – Better Key and Tweak Schedules

We move from Even-Mansour to an Alternating-Key Schedule because:

- Security bounds are better and more "normal" (as already seen).
- Longer tweak ⇒ the adversary has more control on the internal states.
- Hence, we may need more rounds if we kept the Even-Mansour scheme.
- A better key schedule may help balance things.

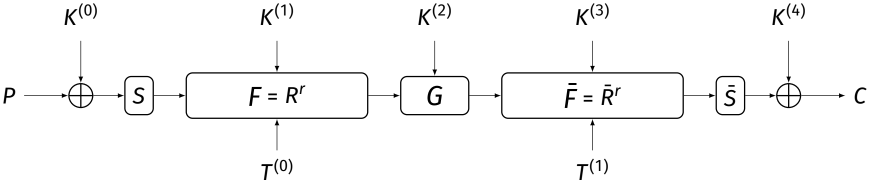
In fact, we shall also see that a better tweak schedule can allow longer tweaks at no extra cost in terms of rounds needed (at least in some cases).

Design

Overall Scheme

Overall Scheme

We keep the *reflector construction*.



Use the same circuit for both encryption and decryption with a minor set-up step.
The first and last rounds consist of a key addition and a S-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.
A bar over a function denotes its inverse, for instance $\bar{R} = R^{-1}$.

Building Blocks

The State

The internal state of the cipher has a size of b bits.

A b-bit value is called a *block*.
It is 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*:

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

Thus, $b = 64\,\ell$.

Both key and tweak have a size of $2\,b = 128\,\ell$ bits.

The State

The internal state of the cipher has a size of b bits.

A b-bit value is called a *block*.
It is 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*:

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

Thus, $b = 64\,\ell$.

Both key and tweak have a size of $2\,b = 128\,\ell$ bits.

The State

The internal state of the cipher has a size of b bits.

A b-bit value is called a *block*.
It is 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*:

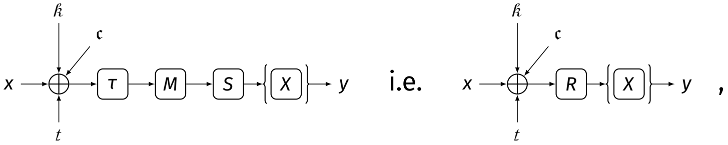
$$L = c_0 \| c_1 \| \cdots \| c_{14} \| c_{15} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{pmatrix} .$$

Thus, $b = 64\,\ell$.

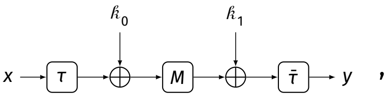
Both key and tweak have a size of $2\,b = 128\,\ell$ bits.

The Round Function and the Reflector

A full round is



where $R = S \circ M \circ \tau$, and X swaps the first and the second row of the first layer with the first and the second row of the second layer (for $\ell = 2$ only). The reflector is



where k_0, k_1 are two round keys.

The State Shuffle (let us just say it is a cell permutation)

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

$$(\pi(L))_i = c_{\pi(i)} \quad \text{for} \quad 0 \leq i < 16 \ .$$

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

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

i.e. it acts on each layer as follows

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

The Diffusion Matrix

Let ρ denote the cyclic rotation to the left of the four bits in a cell, i.e.,

$$\rho(\mathbf{x}) = \rho((x_3, x_2, x_1, x_0)) = \mathbf{x} \lll 1 = (x_2, x_1, x_0, x_3) \ .$$

ρ is a linear map and ρ^4 = identity. The diffusion matrix M is the circulant

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

Involutory. Almost-MDS, like `MIDORI`'s $M_0 := \mathrm{circ}(0, 1, 1, 1)$ and `QARMAv1`'s $M_{4,2} = \mathrm{circ}(0, \rho, \rho^2, \rho)$.

They are grouped into *classes* depending on their transition patterns:
Class I includes M_0 and $M_{4,1}$. $M_{4,2}$ is a *Class II* matrix.

The S-Box

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

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

For certain applications we optionally allow the use of QARMAv1's σ_0.

The road that led to the choice of S-Boxes has been bumpy.
We shall come to it later.

The S-Box

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

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

For certain applications we optionally allow the use of QARMAv1's σ_0.

The road that led to the choice of S-Boxes has been bumpy.
We shall come to it later.

The S-Box

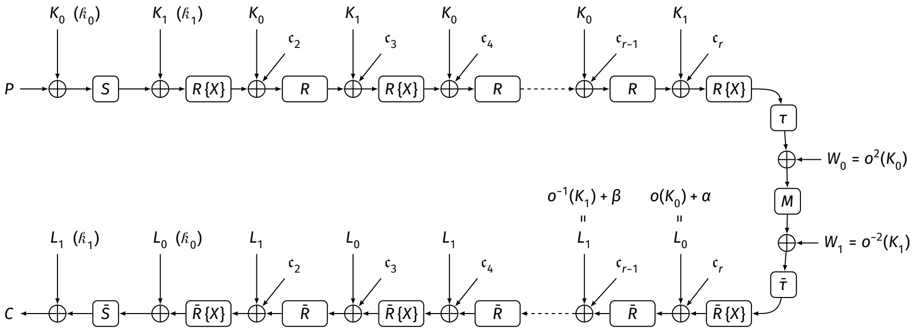
For the general-purpose versions of QARMAv2, we use the following S-Box

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

For certain applications we optionally allow the use of QARMAv1's σ_0.

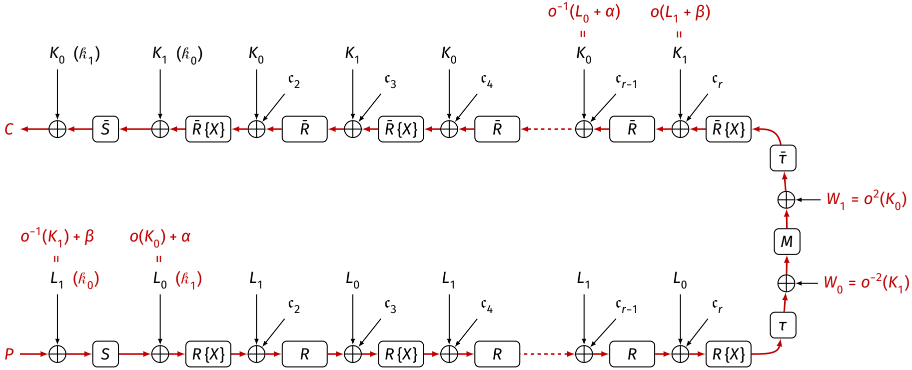
The road that led to the choice of S-Boxes has been bumpy.
We shall come to it later.

Key schedule for Encryption (odd r, ignoring tweaks for now)



Texts / tweak / state = vectors of sixteen or thirty-two cells / 4 by 4 by {1 or 2} tensors
$R = S \circ M \circ \tau$: τ = State Shuffle; M = Involutory Almost MDS; S = 16 ℓ S-Boxes.

Key schedule for Decryption (odd r, ignoring tweaks for now)



Texts / tweak / state = vectors of sixteen or thirty-two cells / 4 by 4 by {1 or 2} tensors
$R = S \circ M \circ \tau$: τ = State Shuffle; M = Involutory Almost MDS; $S = 16\,\ell$ S-Boxes.

The Tweak Schedule

We observe that if we use a fixed permutation to modify the tweak, by continuing with the same transformation through the reflector we are sort of implying that in an attack the schedule must “work well” with the function F and its inverse.

Our goal is to retain some kind of symmetry though.

Hence, we define

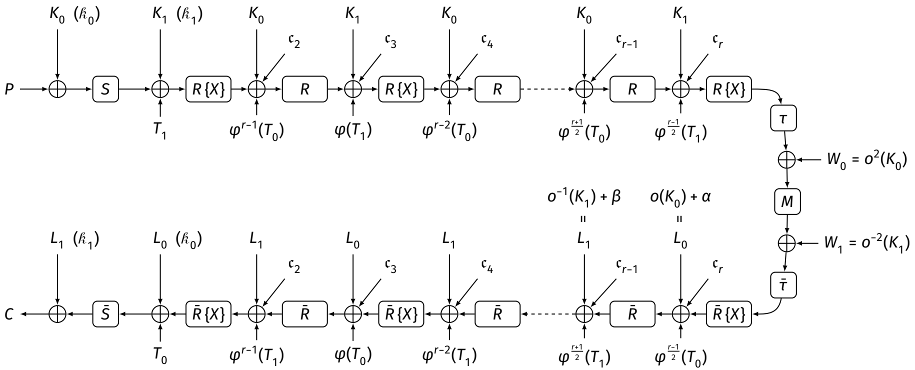
$$\left[\, T_1,\ \varphi^{r-1}(T_0),\ \varphi(T_1),\ \varphi^{r-2}(T_0),\ \varphi^{2}(T_1),\ \varphi^{r-3}(T_0),\ \dots,\ \varphi^{r-1}(T_1),\ T_0 \,\right]\ .$$

Swapping T_0 with T_1 gives the inverse schedule.

We “just” need to find a suitable φ.

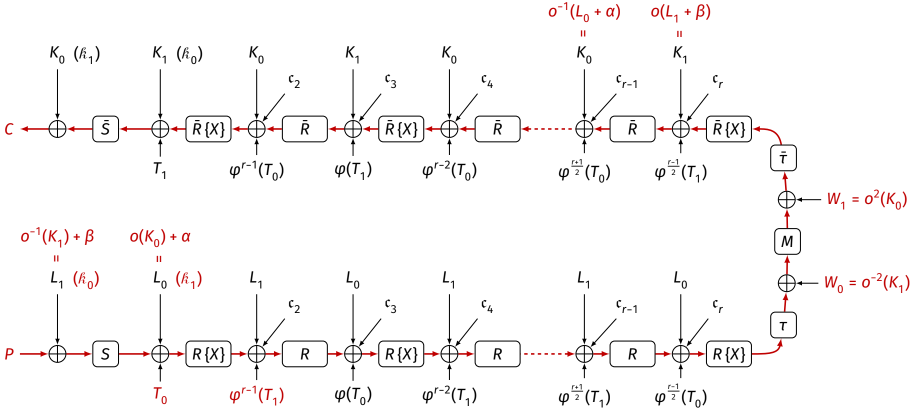
Encryption and Decryption

QARMAv2 Encryption (odd r)



Texts / tweak / state = vectors of sixteen or thirty-two cells / 4 by 4 by {1 or 2} tensors
$R = S \circ M \circ \tau$: τ, φ = State, Tweak Shuffles; M = Involutory Almost MDS; S = 16 ℓ S-Boxes.

QARMAv2 Decryption (odd *r*): using the same circuit



Texts / tweak / state = vectors of sixteen or thirty-two cells / 4 by 4 by {1 or 2} tensors
$R = S \circ M \circ \tau$: τ, φ = State, Tweak Shuffles; M = Involutory Almost MDS; S = 16 ℓ S-Boxes.

Cryptanalysis

arm

Attacks

arm

Estimated reach of various types of cryptanalysis

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

Values are for two independent tweak blocks, except numbers in parentheses, which are specific for a single block tweak, stretched.
* Integral has been recently extended to 10, rep. 11 rounds.

Security claims and parameter choices

With two independent tweak blocks.

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

- Earlier I said "In fact, we shall also see that a better tweak schedule can allow longer tweaks at no extra cost in terms of rounds needed."
  So why are we increasing the number of rounds in some cases?
- The increase in rounds for QARMAv2-64 w.r.t. QARMAv1-64, is only due to boomerang attacks (QARMAv1/MANTIS "borrowed" from MIDORI).

Security claims and parameter choices

With two independent tweak blocks.

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

- Earlier I said "In fact, we shall also see that a better tweak schedule can allow longer tweaks at no extra cost in terms of rounds needed."
  So why are we increasing the number of rounds in some cases?

- The increase in rounds for QARMAv2-64 w.r.t. QARMAv1-64, is only due to boomerang attacks (QARMAv1/MANTIS "borrowed" from MIDORI).

Focus 1: Invariant cosets, the S-Box and the linear layer

arm

Choice of the S-Boxes

As in QARMAv1, the S-Boxes are selected using heuristics.

- Optimal cryptographic properties (DU = 4, Lin = 8).
- All 15 non-zero component functions have algebraic degree 3.
  (σ_0 was an exception here.)
- Each input bit perturbs (i.e. influences non-linearly) all output bits.
  (σ_0 was again an exception here.)
- And some limits on the weights of QMC SOP/NOT-SOP and ANF, which ...
- ... in practice lead to almost optimal depth (3.5 or 4 GE).
  (Verified with Qiao Kexin's database of low-latency boolean functions:
  `https://github.com/qiaokexin/Sbox-depth-evaluation`.)

arm

Enter the linear layer

- An early QARMAv2 used σ_1 with $M_{4,1}$.
- Tim Beyne found non-linear invariants. Unlikely this gives weak keys, but...
- Reverting to QARMAv1's $M_{4,2}$ would have avoided them, but could also make many other things worse.
  Using σ_0 also fine, but algebraic degree not always maximal.
- So we add a subspace chain analysis and a generalized (multi-round) invariant check to our S-Box search.

The key argument in a nutshell (maybe skip during talk)

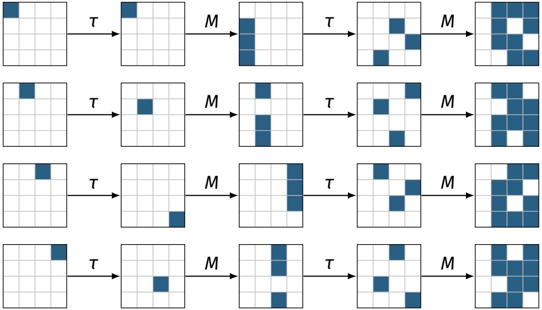
The considered subsets are cosets of linear subspaces.
Dimension 0 and 1 addressed by linear and differential cryptanalysis more generally.
The mappings of dimension 2 cosets under $M_{4,1}$ and ק induce following maps of subspaces:

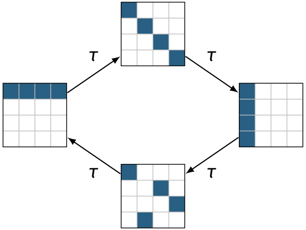
$$
\begin{cases}
\langle 2, \mathrm{C}\rangle, \langle 1, 6\rangle, \langle 3, 8\rangle \xmapsto{\rho^{\{1,2,3\}}} \langle 4, 9\rangle \xmapsto{\text{ק}} \langle 3, 4\rangle \\
\langle 2, 4\rangle, \langle 1, 2\rangle, \langle 1, 8\rangle \xmapsto{\rho^{\{1,2,3\}}} \langle 4, 8\rangle \xmapsto{\text{ק}} \langle 4, 8\rangle \\
\langle 1, 4\rangle, \langle 2, 8\rangle, \langle 1, 4\rangle \xmapsto{\rho^{\{1,2,3\}}} \langle 2, 8\rangle \xmapsto{\text{ק}} \langle 5, 9\rangle \\
\langle 7, \mathrm{A}\rangle, \langle 5, \mathrm{B}\rangle, \langle 7, \mathrm{A}\rangle \xmapsto{\rho^{\{1,2,3\}}} \langle 5, \mathrm{B}\rangle \xmapsto{\text{ק}} \langle 6, \mathrm{B}\rangle \\
\langle 5, \mathrm{B}\rangle, \langle 7, \mathrm{A}\rangle, \langle 5, \mathrm{B}\rangle \xmapsto{\rho^{\{1,2,3\}}} \langle 7, \mathrm{A}\rangle \xmapsto{\text{ק}} \langle 7, 9\rangle
\end{cases}
$$

and all other ones map to dimension 3 or 4. Dimension 3 cosets all map to dimension 4 ones. Then: distinguishers using subspace trails are at most 5, resp. 7 rounds (Three rounds as Dim 1 $\mapsto$ $\mapsto$ Dim 2 $\mapsto$ Dim 2 $\mapsto$ Dim 3, then add max 2 + 0, resp. 3 + 1 rounds for $\ell = 1, 2$ before + after). QED.

Focus 2: Finding better tweak schedules

Finding tweak shuffles – Main argument: Cancellation





Use avoidance of self-cancellations as a starting point, then fine-tune.

First consider τ^2. Then apply row permutations and an additional swap involving non affected cells to get maximal cyclic order 16.

And then active S-Box counts (cell-wise MILP model)

With two independent tweak blocks.

		Half-Cipher										Full-Cipher					
	r =	3	4	5	6	7	8	9	10	11	12	2	3	4	5	6	7
ℓ	*Rounds* =	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*6*	*8*	*10*	*12*	*14*	*16*
1	RT Diff.	2	4	8	12	16	22	24	27	32	36	5	12	24	32	41	52
	Linear	16	23	30	35	38	41	50	57	62	67	5	32	50	64	72	–
2	RT 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*	–

With a single block tweak.

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

QARMAv1.

1	RT Diff.	6	10	16	21	24	27	32	36	40	45	6	14	24	32	42	52

And then active S-Box counts (cell-wise MILP model)

With two independent tweak blocks.

		Half-Cipher										Full-Cipher					
	r =	3	4	5	6	7	8	9	10	11	12	2	3	4	5	6	7
ℓ	*Rounds* =	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*6*	*8*	*10*	*12*	*14*	*16*
1	RT Diff.	2	4	8	12	16	22	24	27	32	36	5	12	24	32	41	52
	Linear	16	23	30	35	38	41	50	57	62	67	5	32	50	64	72	–
2	RT 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*	–

With a single block tweak.

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

QARMAv1.

1	RT Diff.	6	10	16	21	24	27	32	36	40	45	6	14	24	32	42	52

arm

And then active S-Box counts (cell-wise MILP model)

With two independent tweak blocks.

		Half-Cipher										Full-Cipher					
	r =	3	4	5	6	7	8	9	10	11	12	2	3	4	5	6	7
ℓ	*Rounds* =	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*6*	*8*	*10*	*12*	*14*	*16*
1	RT Diff.	2	4	8	12	16	22	24	27	32	36	5	12	24	32	41	52
	Linear	16	23	30	35	38	41	50	57	62	67	5	32	50	64	72	–
2	RT 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*	–

With a single block tweak.

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

QARMAv1.

1	RT Diff.	6	10	16	21	24	27	32	36	40	45	6	14	24	32	42	52

Focus 3: Almost MDS has better diffusion than MDS …

(skip during talk – good for afternoon?)

WIP – Almost MDS has better diffusion than MDS!

... for TBCs (maybe). Active S-Box counts in related-tweak differential characteristics.

Rounds	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
For an AES-like round with a DEOXYS-like tweak schedule.															
k^*_{35}	5	7	11	17	20	25	28	33	37	39	43	48	53	57	60
For a MIDORI-like round with MIDORI's τ and a QARMAv1-like tweak schedule.															
h	3	6	10	16	21	24	27	32	36	40	45	49	54	57	60
τ_{h4}	3	6	10	14	19	24	28	32	37	42	46	50	54	58	63

Counts similar, but an Almost-MDS matrix is lighter and has shorter critical path.
Hence, a design with a similar number of rounds is also faster. TBD: full-cipher counts.

WIP – Almost MDS has better diffusion than MDS!

... for TBCs (maybe). Active S-Box counts in related-tweak differential characteristics.

Rounds	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
For an AES-like round with a DEOXYS-like tweak schedule.															
k^*_{35}	5	7	11	17	20	25	28	33	37	39	43	48	53	57	60
For a MIDORI-like round with MIDORI's τ and a QARMAv1-like tweak schedule.															
h	3	6	10	16	21	24	27	32	36	40	45	49	54	57	60
τ_{h4}	3	6	10	14	19	24	28	32	37	42	46	50	54	58	63

Counts similar, but an Almost-MDS matrix is lighter and has shorter critical path.
Hence, a design with a similar number of rounds is also faster. TBD: full-cipher counts.

WIP – Almost MDS has better diffusion than MDS!

... for TBCs (maybe). Active S-Box counts in related-tweak differential characteristics.

Rounds	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
For an AES-like round with a DEOXYS-like tweak schedule.															
k^*_{35}	5	7	11	17	20	25	28	33	37	39	43	48	53	57	60
For a MIDORI-like round with MIDORI's τ and a QARMAv1-like tweak schedule.															
h	3	6	10	16	21	24	27	32	36	40	45	49	54	57	60
τ_{h4}	3	6	10	14	19	24	28	32	37	42	46	50	54	58	63

Counts similar, but an Almost-MDS matrix is lighter and has shorter critical path. Hence, a design with a similar number of rounds is also faster. TBD: full-cipher counts.

Implementation

Implementations (5 nm TSMC, low voltage)

				Area optimized			Latency optimized		
				Area		Delay	Area		Delay
Cipher	*Rounds*	*Tweak*	*Security Claims*	μm^2	GE	ps	μm^2	GE	ps
PRESENT-80	31	N	$D \geq 2^{64} \parallel T \geq 2^{80}$	812.0	10175	1836	1815.7	22752	953
PRESENT-128	31	N	$D \geq 2^{64} \parallel T \geq 2^{128}$	848.8	10636	1841	1824.1	22858	958
MIDORI-64 (*)	16	N	$D \geq 2^{64} \parallel T \geq 2^{128}$	443.5	5557	921	761.8	9546	678
PRINCE	12	N	$D \times T \geq 2^{126}$	334.6	4193	710	672.1	8422	534
SKINNY-64-192 (i.e. $\mathfrak{k} + \mathfrak{t} = 192$)	32	Y	$D \geq 2^{41.5}$ (†)	918.3	11507	1951	1682.0	21078	1254
MANTIS-6	14	Y	$D \times T \geq 2^{126}$	425.4	5331	734	715.8	8969	592
MANTIS-7	16	Y	$D \times T \geq 2^{126}$	485.6	6085	854	788.4	9879	683
MANTIS-8	18	Y	$D \times T \geq 2^{126}$	545.8	6839	974	861.0	10789	774
BIPBIP-Dec (i.e. $b = 24$, $\mathfrak{t} = 40$)	11	Y	$T \geq 2^{72} \parallel D \geq 2^{72} \parallel TD \geq 2^{96}$	303.7	3806	647	381.1	4776	436
BIPBIP-Enc (i.e. $b = 24$, $\mathfrak{t} = 40$)	11	Y	(same)	514.7	6450	1480	1090.3	13662	909
QARMAv1-64-σ_0 ($r = 5$, PAC, $\mathfrak{t} = 64$)	12	Y	$CP \geq 2^{30}$, $KP \geq 2^{40}$	394.7	4946	728	707.0	8860	525
QARMAv1-64 ($r = 7$, $\mathfrak{t} = 64$)	16	Y	$D \times T \geq 2^{126}$	551.7	6913	1030	996.6	12489	731
QARMAv2-64-σ_0 ($r = 4$, PAC $\leq$ 10 bits)	10	Y	$T \approx 2^{128}$	309.7	3881	606	495.9	6214	430
QARMAv2-64-σ_0 ($r = 5$, PAC $\leq$ 24 bits)	12	Y	$T \approx 2^{128}$	374.6	4694	721	600.8	7529	514
QARMAv2-64-σ_0 ($r = 6$, PAC $\leq$ 48 bits)	14	Y	$T \approx 2^{128}$	435.4	5456	829	721.2	9038	600
QARMAv2-64 ($r = 7$, $\mathfrak{t} = 64$)	16	Y	$D \geq 2^{56} \parallel T \geq 2^{128}$	537.0	6729	936	954.4	11959	706
QARMAv2-64 ($r = 9$, $\mathfrak{t} = 128$)	20	Y	$D \geq 2^{56} \parallel T \geq 2^{128}$	675.2	8461	1173	1187.3	14879	885

$\mathfrak{k}$, $\mathfrak{t}$ = size of key, resp. tweak in bits. (*) = we include MIDORI-64 because it could be easily repaired. (†) = inferred from original analysis.

Implementations (5 nm TSMC, low voltage)

Cipher	*Rounds*	*Tweak*	*Security Claims*	*Area optimized*			*Latency optimized*		
				Area		Delay	Area		Delay
				μm^2	GE	ps	μm^2	GE	ps
AES-128	10	N	$D \geq 2^{128} \parallel T \geq 2^{128}$	2304.1	28873	3064	4520.6	56648	1791
AES-192	12	N	$D \geq 2^{128} \parallel T \geq 2^{128}$	2635.4	33025	3686	5023.6	62952	2153
AES-256	14	N	$D \geq 2^{128} \parallel T \geq 2^{128}$	3238.7	40585	4290	6191.5	77587	2513
MIDORI-128	20	N	$D \geq 2^{128} \parallel T \geq 2^{128}$	1085.1	13597	1156	1954.5	24492	840
ASCON-p^{12} (note: b = 320)	12	N	$D \geq 2^{64} \parallel T \geq 2^{128}$	2228.3	27923	826	2766.8	34671	507
SPEEDY-5 (note: b = 192)	5	N	$D \geq 2^{64} \parallel T \geq 2^{128}$	1571.8	18567	650	2668.0	33433	384
SPEEDY-7 (note: b = 192)	7	N	$D \geq 2^{128} \parallel T \geq 2^{128}$ (*)	2109.2	26431	924	3599.6	45107	552
SPEEDY-8 (note: b = 192)	8	N	$D \geq 2^{128} \parallel T \geq 2^{192}$ (*)	2423.2	30363	1061	4065.4	50944	636
SKINNY-128-128 (i.e. $\mathfrak{k} + \mathfrak{t}$ = 128)	40	Y	$D \geq 2^{88.5}$ (†)	3986.3	49953	4371	9241.0	115800	2164
SKINNY-128-384 (i.e. $\mathfrak{k} + \mathfrak{t}$ = 384)	40	Y	$D \geq 2^{88.5}$ (†)	4513.6	56560	4348	9527.5	11939	2177
QARMAv1-128 (r = 9, $\mathfrak{t}$ = 128)	20	Y	$D \times T \geq 2^{254}$ (‡)	1422.3	17823	1290	2535.8	31776	912
QARMAv1-128 (r = 11, $\mathfrak{t}$ = 128)	24	Y	$D \times T \geq 2^{254}$	1635.6	20496	1561	3078.3	38575	1091
QARMAv2-128-128 (r = 9, $\mathfrak{t}$ = 128)	20	Y	$D \geq 2^{80} \parallel T \geq 2^{128}$	1347.5	16886	1170	2337.5	29292	890
QARMAv2-128-128 (r = 11, $\mathfrak{t}$ = 256)	24	Y	$D \geq 2^{80} \parallel T \geq 2^{128}$	1620.3	20305	1409	2875.8	36037	1068
QARMAv2-128-192 (r = 13, $\mathfrak{t}$ = 256)	28	Y	$D \geq 2^{80} \parallel T \geq 2^{192}$	1893.5	23727	1645	3333.0	41778	1248
QARMAv2-128-256 (r = 15, $\mathfrak{t}$ = 256)	32	Y	$D \geq 2^{80} \parallel T \geq 2^{256}$	2166.8	27152	1879	3797.8	47592	1425

$\mathfrak{k}$, $\mathfrak{t}$ = size of key, resp. tweak in bits. (*) = Estimated by us on the basis of cryptanalysis, adding one round to the original claims.
(†) = inferred from original analysis. (‡) = Tweak masking suggested.

Questions

THANK YOU!

