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

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!

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^{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	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*	4 + 6	Y	$2^{237.3}$	2^{122} CP	2^{144}	Trunc. imp. diff.	[YQC18]
128	2 + 8	Y	$2^{120.94}$	$2^{104.02}$ CP	$2^{94.50}$	Rel-tweak Imp. Diff.	[DuWLW22]
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]
128	8 + 3	Y	$2^{104.60}$	$2^{124.05}$ CP	2^{48}	Rel-tweak trunc. diff.	[SII23]
128	9 + 4	Y	$2^{238.02}$	$2^{106.63}$ CP	2^{240}	Rel-tweak trunc. diff.	[SII23]

Why QARMAv2?

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

  During the last seven years we achieved a better understanding of block cipher design, and of the requirements coming from practical applications.

- Longer tweaks to address use cases and for better security.

- Revised components to improve security.

Why QARMAv2?

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

  During the last seven years we achieved a better understanding of block cipher design, and of the requirements coming from practical applications.

- Longer tweaks to address use cases and for better security.

- Revised 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)

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

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

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

$$\text{if Memory} \leq 2^{56 \text{ resp. } 80}, \text{ then Time} \geq 2^{\text{key size}}$$

similarly to `PRINCEv2`.

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

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

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

$$\boxed{\text{if Memory} \leq 2^{56 \text{ resp. } 80}\text{, then Time} \geq 2^{\text{key size}}}$$

similarly to PRINCEv2.

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 QARMAv1-128 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.

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 QARMAv1-128 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.

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 QARMAv1-128 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

  may 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.
- 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 QARMAv1-128 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

  may 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.
- For embedded: 64-bit blocks, and 128-bit keys and tweaks should be ok.

Security — Better Key and Tweak Schedules

- With a TBC, key changed infrequently. We do not consider related-key attacks.
- Tweak changes often, Adversary may control it. Consider related-tweak attacks.
- So, we do not consider a “tweakey”, but rather tweak and key separately.
- We move from Even-Mansour to an Alternating-Key Schedule because:
- Security bounds are better and more “normal” (as already seen).
- OTOH longer tweak $\Rightarrow$ the adversary has more control.
- Hence, we may need more rounds if we kept the Even-Mansour scheme.
- Better key/tweak schedule may help offset this.

Security — Better Key and Tweak Schedules

- With a TBC, key changed infrequently. We do not consider related-key attacks.
- Tweak changes often, Adversary may control it. Consider related-tweak attacks.
- So, we do not consider a “tweakey”, but rather tweak and key separately.
- We move from Even-Mansour to an Alternating-Key Schedule because:
- Security bounds are better and more “normal” (as already seen).
- OTOH longer tweak ⇒ the adversary has more control.
- Hence, we may need more rounds if we kept the Even-Mansour scheme.
- Better key/tweak schedule may help offset this.

Security — Better Key and Tweak Schedules

- With a TBC, key changed infrequently. We do not consider related-key attacks.
- Tweak changes often, Adversary may control it. Consider related-tweak attacks.
- So, we do not consider a “tweakey”, but rather tweak and key separately.
- We move from Even-Mansour to an Alternating-Key Schedule because:
- Security bounds are better and more “normal” (as already seen).
- OTOH longer tweak ⇒ the adversary has more control.
- Hence, we may need more rounds if we kept the Even-Mansour scheme.
- Better key/tweak schedule may help offset this.

arm

Security — Better Key and Tweak Schedules

- With a TBC, key changed infrequently. We do not consider related-key attacks.
- Tweak changes often, Adversary may control it. Consider related-tweak attacks.
- So, we do not consider a “tweakey”, but rather tweak and key separately.
- We move from Even-Mansour to an Alternating-Key Schedule because:
- Security bounds are better and more “normal” (as already seen).
- OTOH longer tweak $\Rightarrow$ the adversary has more control.
- Hence, we may need more rounds if we kept the Even-Mansour scheme.
- Better key/tweak schedule may help offset this.

Security — Better Key and Tweak Schedules

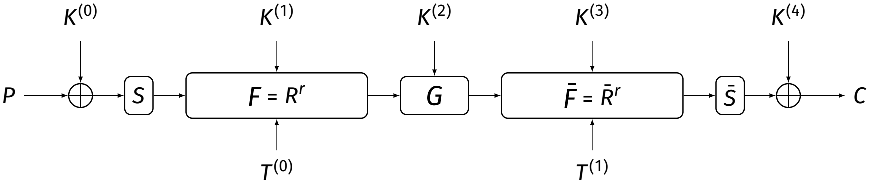
- With a TBC, key changed infrequently. We do not consider related-key attacks.
- Tweak changes often, Adversary may control it. Consider related-tweak attacks.
- So, we do not consider a “tweakey”, but rather tweak and key separately.
- We move from Even-Mansour to an Alternating-Key Schedule because:
- Security bounds are better and more “normal” (as already seen).
- OTOH longer tweak ⇒ the adversary has more control.
- Hence, we may need more rounds if we kept the Even-Mansour scheme.
- Better key/tweak schedule may help offset this.

Design

arm

Overall Scheme

Overall Scheme: Keep the *Reflector Construction*



Use the same circuit for both encryption and decryption with a minor set-up step.

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*:

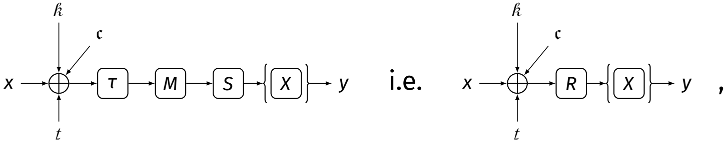
$$L = c_0 \| c_1 \| \cdots \| c_{14} \| c_{15} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{pmatrix} .$$

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

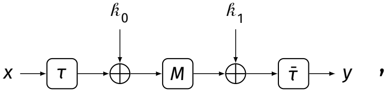
Both key and tweak have a size of $2\,b = 128\,\ell$ bits.

The Round Function and the Reflector

A full round is



where $R = S \circ M \circ \tau$, and X swaps the first two rows between the two layers (for $\ell = 2$ only). τ is the same cell shuffle used in `MIDORI`, `MANTIS`, and `QARMAv1`. The reflector is



where k_0, k_1 are two round keys.

The State (Cellwise) Shuffle

The `MIDORI` *state shuffle*

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

acts on each layer cellwise 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 linear, 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 $\mathrm{circ}(0, 1, 1, 1)$ and `QARMAv1`'s $\mathrm{circ}(0, \rho, \rho^2, \rho)$.

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 PAC we allow the use of QARMAv1's σ_0.)

The road that led to the choice of S-Boxes has been bumpy.

We changed S-Box because Tim Beyne found some invariants if the new matrix is used with the old S-Box. (The TL;DR is: stricter filtering in S-Box search + new analysis of propagation of affine subspaces.)

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.

Hence, we define

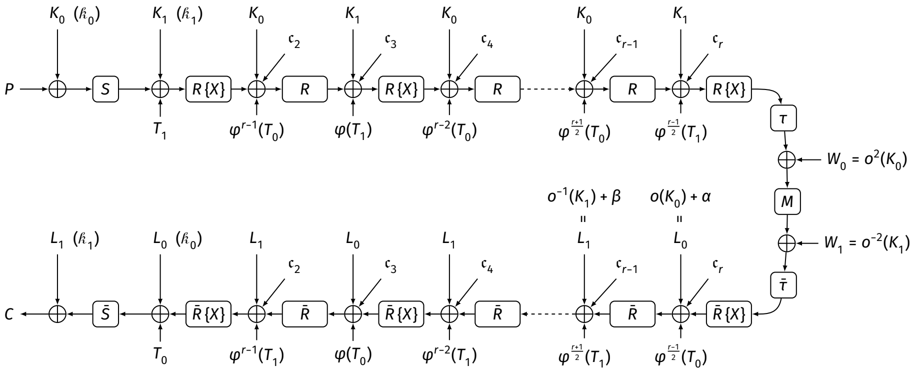
$$\left[\, T_1,\ \varphi^{r-1}(T_0),\ \varphi(T_1),\ \varphi^{r-2}(T_0),\ \varphi^2(T_1),\ \varphi^{r-3}(T_0),\ \dots\ ,\ \varphi^{r-1}(T_1),\ T_0 \,\right]\ .$$

Swapping T_0 with T_1 gives the inverse schedule.
(Some symmetry necessary to allow easy setup.)

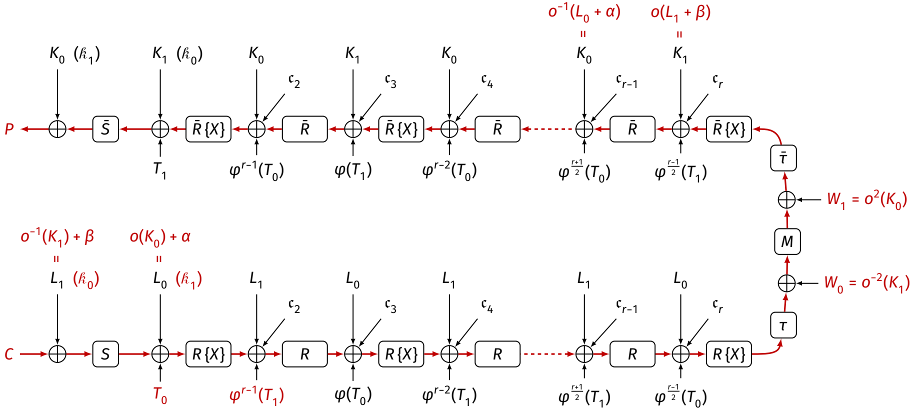
We “just” need to find a suitable φ.

Encryption
&
Decryption

QARMAv2 Encryption (odd r)



QARMAv2 Decryption (odd r): using the same circuit



Security

arm

Cryptanalysis

Estimated reach of various types of cryptanalysis

Attack	QARMAv2-64 Parameter *r*	QARMAv2-64 Rounds	QARMAv2-128 Parameter *r*	QARMAv2-128 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

- A goal was to **not** increase the number of rounds.
- This was not achieved for QARMAv2-64.
- The reason is: Boomerang attacks.

Security claims and parameter choices

With two independent tweak blocks.

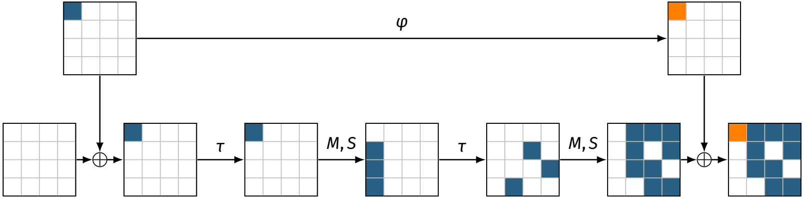
Variant	Block Size	Key Size	Time	Data	Parameter	Rounds
QARMAv2-64-128	64 bits	128 bits	$2^{128-\varepsilon}$	2^{56}	$r = 9$	20
QARMAv2-128-128	128 bits	128 bits	$2^{128-\varepsilon}$	2^{80}	$r = 11$	24
QARMAv2-128-192	128 bits	192 bits	$2^{192-\varepsilon}$	2^{80}	$r = 13$	28
QARMAv2-128-256	128 bits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

- A goal was to **not** increase the number of rounds.
- This was not achieved for QARMAv2-64.
- The reason is: Boomerang attacks.

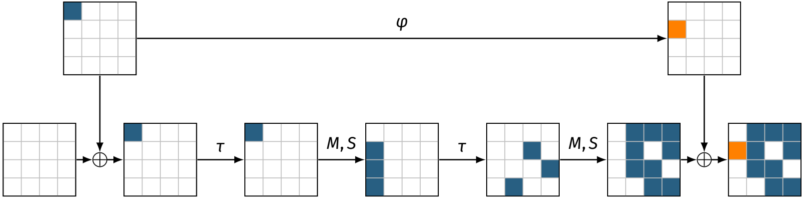
Focus 1: Finding better tweak schedules

arm

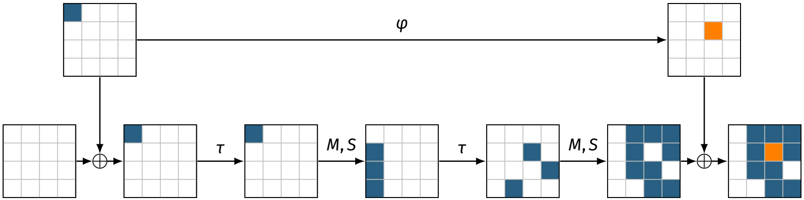
Finding tweak shuffles — Main idea: avoid self-cancellation



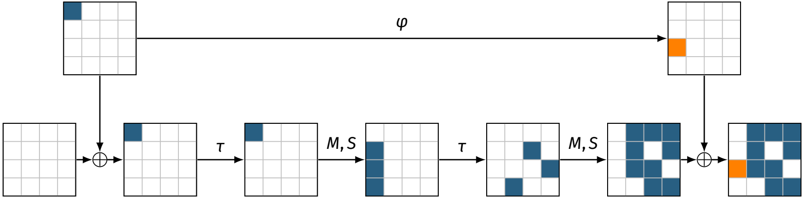
Finding tweak shuffles — Main idea: avoid self-cancellation



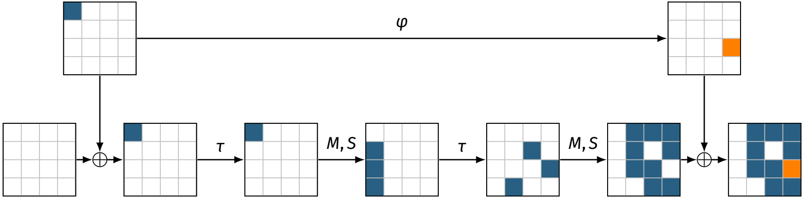
Finding tweak shuffles — Main idea: avoid self-cancellation



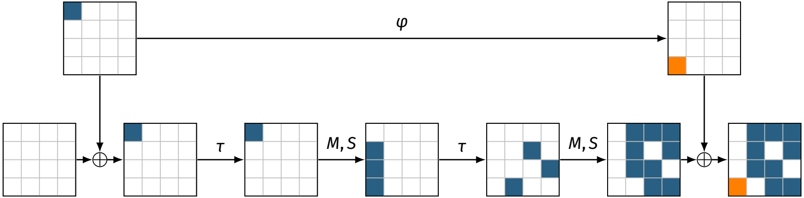
Finding tweak shuffles — Main idea: avoid self-cancellation



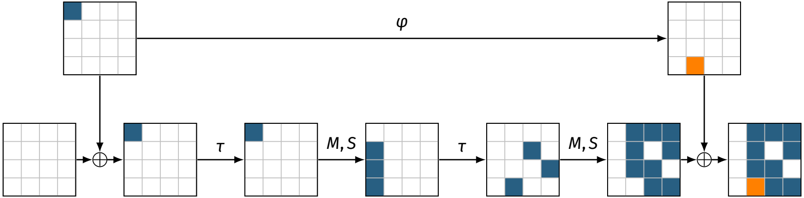
Finding tweak shuffles — Main idea: avoid self-cancellation



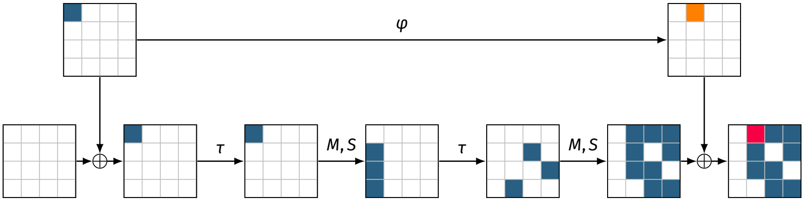
Finding tweak shuffles — Main idea: avoid self-cancellation



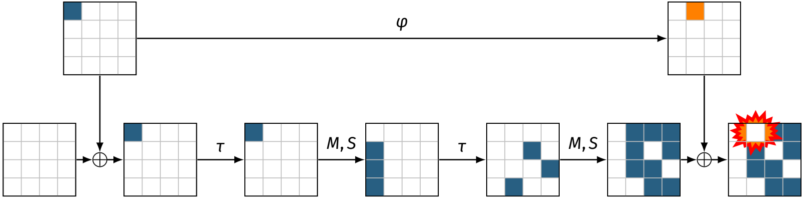
Finding tweak shuffles — Main idea: avoid self-cancellation



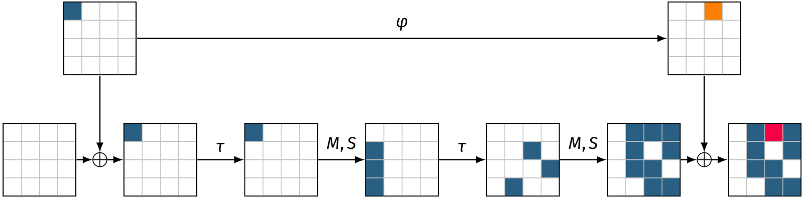
Finding tweak shuffles — Main idea: avoid self-cancellation



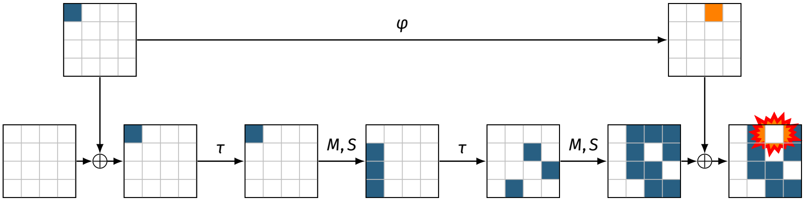
Finding tweak shuffles — Main idea: avoid self-cancellation



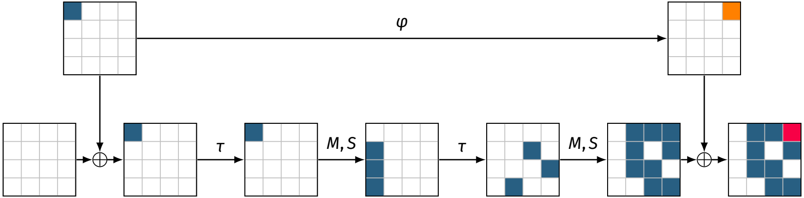
Finding tweak shuffles — Main idea: avoid self-cancellation



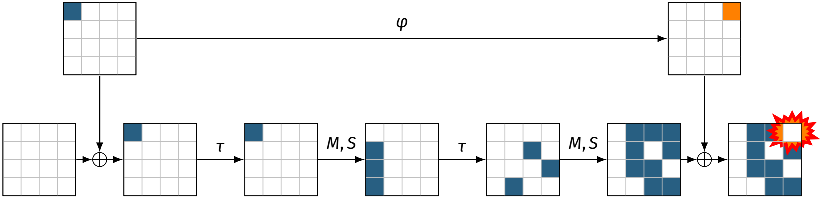
Finding tweak shuffles — Main idea: avoid self-cancellation



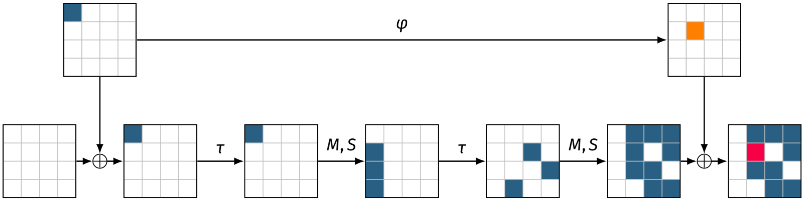
Finding tweak shuffles — Main idea: avoid self-cancellation



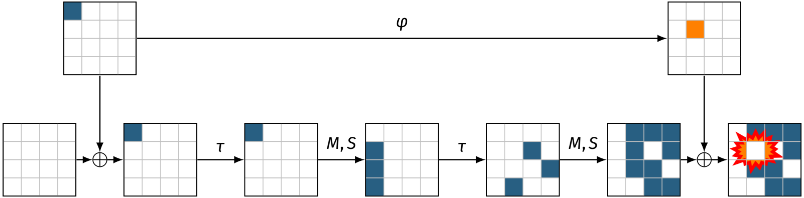
Finding tweak shuffles — Main idea: avoid self-cancellation



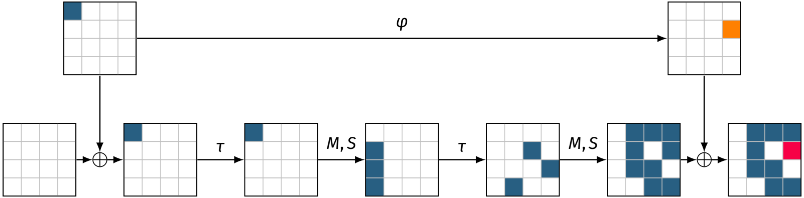
Finding tweak shuffles — Main idea: avoid self-cancellation



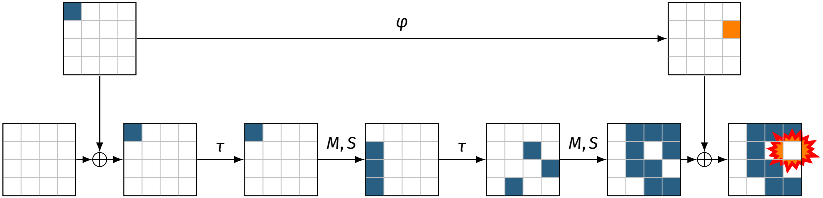
Finding tweak shuffles — Main idea: avoid self-cancellation



Finding tweak shuffles — Main idea: avoid self-cancellation

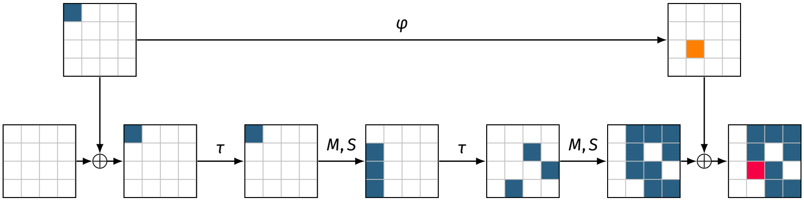


Finding tweak shuffles — Main idea: avoid self-cancellation

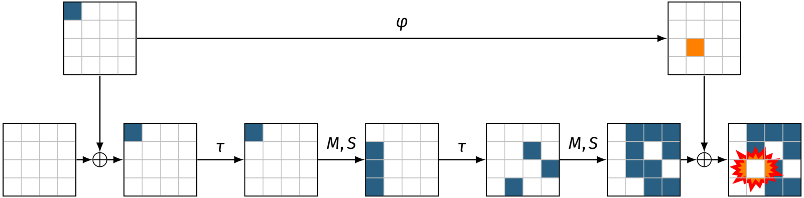


arm

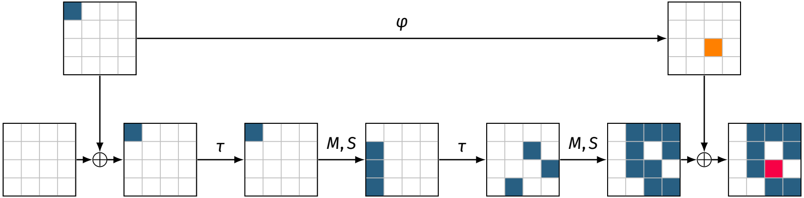
Finding tweak shuffles — Main idea: avoid self-cancellation



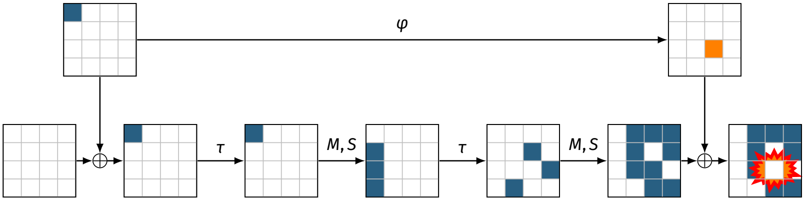
Finding tweak shuffles — Main idea: avoid self-cancellation



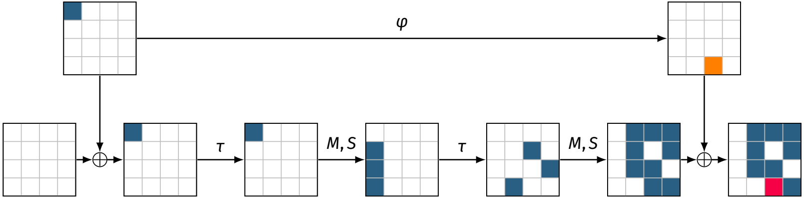
Finding tweak shuffles — Main idea: avoid self-cancellation



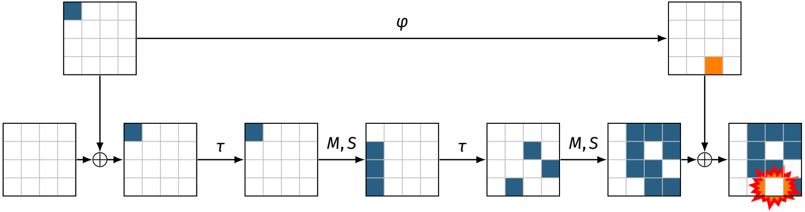
Finding tweak shuffles — Main idea: avoid self-cancellation



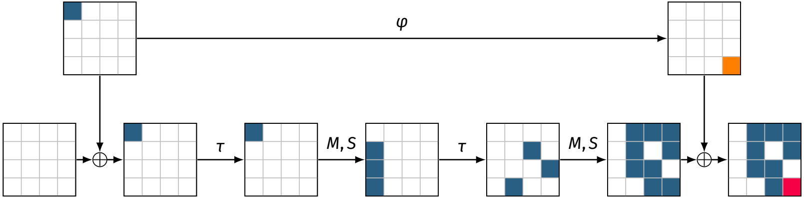
Finding tweak shuffles — Main idea: avoid self-cancellation



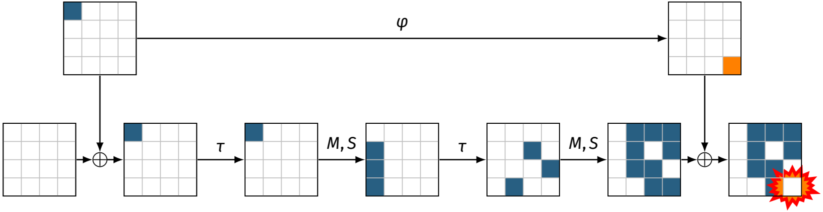
Finding tweak shuffles — Main idea: avoid self-cancellation



Finding tweak shuffles — Main idea: avoid self-cancellation

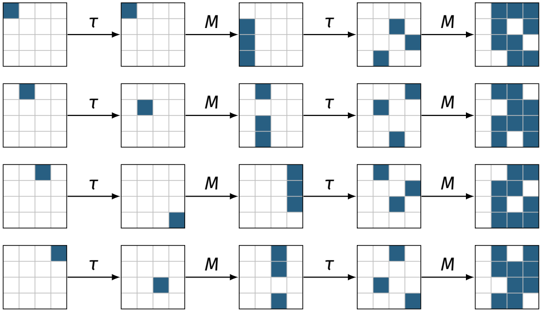


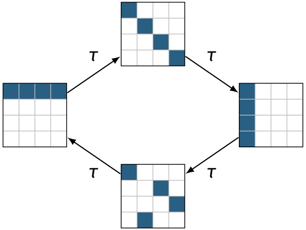
Finding tweak shuffles — Main idea: avoid self-cancellation



arm

Finding tweak shuffles — Main idea: avoid self-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)

QARMAv2

ℓ	r =	2	3	4	5	6	7
	Rounds =	*6*	*8*	*10*	*12*	*14*	*16*
1	RT Diff.	5	12	24	32	41	52
	Linear	5	32	50	64	72	–
2	RT Diff.	5	16	32	52	*61*	–
	Linear	24	44	56	80	*96*	–

QARMAv1

1	RT Diff.	6	14	24	32	42	52

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

QARMAv2

ℓ							
	r =	2	3	4	5	6	7
	Rounds =	*6*	*8*	*10*	*12*	*14*	*16*
1	RT Diff.	5	12	24	32	41	52
	Linear	5	32	50	64	72	–
2	RT Diff.	5	16	32	52	*61*	–
	Linear	24	44	56	80	*96*	–

QARMAv1

1	RT Diff.	6	14	24	32	42	52

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

QARMAv2

ℓ	r = *Rounds* =	2 *6*	3 *8*	4 *10*	5 *12*	6 *14*	7 *16*
1	RT Diff.	5	12	24	32	41	52
	Linear	5	32	50	64	72	–
2	RT Diff.	5	16	32	52	*61*	–
	Linear	24	44	56	80	*96*	–

QARMAv1

1	RT Diff.	6	14	24	32	42	52

arm

Remark on the new schedules

- If you want to keep using QARMAv1, update it with the new key schedule.
- All the cryptanalysis on QARMAv1 should still apply, likely with no smaller complexity.
- If you like the new tweak schedule, go to QARMAv2.

Focus 2: Implementation

Implementations (5 nm TSMC, low voltage)

Cipher	Rounds	Tweak	Security Claims	Area optimized: Area μm^2	Area optimized: Area GE	Area optimized: Delay ps	Latency optimized: Area μm^2	Latency optimized: Area GE	Latency optimized: Delay ps
PRESENT-128	31	N	$D \geq 2^{64} \parallel T \geq 2^{128}$	848.8	10636	1841	1824.1	22858	958
PRINCE	12	N	$D \times T \geq 2^{126}$	334.6	4193	710	672.1	8422	534
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
BIPBIP-Dec (i.e. $b = 24$, $\mathfrak{t} = 40$)	11	Y	$T \gtrsim 2^{72} \parallel D \gtrsim 2^{72} \parallel TD \gtrsim 2^{96}$	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 ($r = 3$, PAC, $\mathfrak{t} = 64$)	8	Y	$CP \geq 2^{20}$, $KP \geq 2^{40}$	251.2	3147	464	450.0	5638	334
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 ≤ 10 bits)	10	Y	$T \approx 2^{128}$	309.7	3881	606	495.9	6214	430
QARMAv2-64-σ_0 ($r = 5$, PAC ≤ 24 bits)	12	Y	$T \approx 2^{128}$	374.6	4694	721	600.8	7529	514
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.

arm

Implementations (5 nm TSMC, low voltage)

Cipher	Rounds	Tweak	Security Claims	Area optimized: Area μm^2	Area optimized: Area GE	Area optimized: Delay ps	Latency optimized: Area μm^2	Latency optimized: Area GE	Latency optimized: Delay ps
AES-128	10	N	$D \geq 2^{128} \parallel T \geq 2^{128}$	2304.1	28873	3064	4520.6	56648	1791
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-6 (note: $b = 192$)	6	N	$D \geq 2^{128} \parallel T \geq 2^{128}$	1795.5	22499	787	3133.8	39270	468
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
QARMAv2-128-256 ($r = 15$, $\mathfrak{t} = 256$)	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.
(†) = inferred from original analysis. (‡) = Tweak masking suggested.

Final Words

See you at FSE 2031 for QARMAv3!!!

THANK YOU!

