The QARMA Chakra

Roberto Avanzi
Architecture and Technology Group
roberto.avanzi@arm.com

Bar Ilan University – 2019 LCD – March 31st, 2019

© ARM 2019
Acknowledgements

What follows is not only my work.

In fact, quite a few parts of it are joint work.

Some of the folks that suffered greatly while working with me are Subhadeep Banik, Light Darkman, Senyang Huang, Francesco Regazzoni, and Andrey Bogdanov.
Acknowledgements

Thanks also to: Can Acar, Satish Anand, Christof Beierle, Christina Boura, Antonio Cardoso Costa, Mike Campbell, Alexander Dent, Simonetta Diaz, Itai Dinur, Xiaoyang Dong, Maria Eichlseder, Alexandr Gantman, Renwei Ge, Scott McGregor, Richard Grisenthwaite, Yatin Hoskote, Jérémy Jean, Gregor Leander, Jason Parker and the amazing, spectacular REDACTED team he leads at ARM, Rene Peralta, Prakash Ramrakhyani, Andrew Rose, Greg Rose, Brian Rosenberg, Andreas Sandberg, David Schall, Arrigo Triulzi, Meltem Sönmez Turan, and Robert Turner.

We used tools developed by: Leo Perrin; Oleksandr Kazymyrov, Maksim Storetvedt, and Anna Maria Eilertsen; Stjepan Picek, Lejla Batina, Domagoj Jakobović, Barış Ege, and Marin Golub; Gregor Leander, Brice Minaud, and Sondre Rønjom; and the SageMath and GUROBI developers.
Connections
A set of technologies developed *around* QARMA, connected with each other and to QARMA itself.
Feature Overview
... a cipher partly designed on the slopes of Mount Carmel ...

Qualcomm

ARM

Authenticator

Q + A + R + M + A

Roberto M. Avanzi

(and it might badly affect my karma)
QARMA Encryption

Texts / tweak / state = vectors of sixteen 4/8-bit cells / $4 \times 4$ matrices

$\tau, h = \text{Cell Shuffles}; M = \text{Involuntary Almost MDS matrix}; S = 16 \text{ S-Boxes}; \omega = \text{LSFR } \times 7$
QARMA Encryption

Texts / tweak / state = vectors of sixteen 4/8-bit cells / 4 × 4 matrices
τ, h = Cell Shuffles; M = Involutory Almost MDS matrix; S = 16 S-Boxes; ω = LSFR × 7
QARMA Decryption

Texts / tweak / state = vectors of sixteen 4/8-bit cells / $4 \times 4$ matrices

$\tau, h =$ Cell Shuffles; $M =$ Involutory Almost MDS matrix; $S =$ 16 S-Boxes; $\omega =$ LSFR $\times$ 7

Decrypt with: $k^0 \mapsto k^0 \oplus \alpha$, swap $w^0$ and $w^1$
QARMA Decryption

Texts / tweak / state = vectors of sixteen 4/8-bit cells / 4 × 4 matrices

\( \tau, h = \text{Cell Shuffles}; M = \text{Involutory Almost MDS matrix}; S = 16 \text{ S-Boxes}; \omega = \text{LSFR} \times 7 \)

Decrypt with: \( k^0 \mapsto k^0 \oplus \alpha \), swap \( w^0 \) and \( w^1 \), replace \( k^1 \mapsto M \cdot k^1 \)
QARMA Encryption

Texts / tweak / state = vectors of sixteen 4/8-bit cells / $4 \times 4$ matrices

$\tau$, $h$ = Cell Shuffles; $M$ = Involutory Almost MDS matrix; $S$ = 16 S-Boxes; $\omega$ = LSFR $\times$ 7

Decrypt with: $k^0 \leftrightarrow k^0 \oplus \alpha$, swap $w^0$ and $w^1$, replace $k^1 \leftrightarrow M \cdot k^1$
Features

- QARMA-64 and QARMA-128 are Public Domain!
- Builds on very well understood design methodologies – all aspects explained and verifiable.
- Designed 18 years after the AES, 5 years after PRINCE, learns from Even-Mansour, MIDORI.
- Stealthily already deployed (Pointer Authentication) – ARM partners already know how to implement it!
- Tweakable. Ideal for ΘCB-like modes with BBB security.
- Very short latency (best in class).
- Area and power consumption not minimal, but very small. Good for parallelism.
- In HW, round probably the lightest among those with a full layer of optimal 4-bit S-Boxes with full diffusion and all non-zero component functions of degree 3, and Almost-MDS diffusion.
- Various pipelining and/or unrolling options possible, including very fine grained.
- Code size: even a t-box implementation should be smaller than the AES’s.
- Mostly nibble and byte based, so implementation easy on microcontrollers.
- Cool name.
- Hopefully, also secure (so far so good).
Considered attacks

- Linear and differential cryptanalysis (MILP models, following Beierle)
- —, under related tweak model (MILP models, following Beierle)
- Reflection Attacks (follows from structure)
- Generic attacks on Even-Mansour schemes (follows from structure)
- Slide attacks (follows from round heterogeneity)
- Meet-in-the-middle attacks (following MIDORI/MANTIS)
- Invariant subspace attacks (heuristic arguments)
- Algebraic cryptanalysis (equations and variables counting, degree growth)
- Imp. diff. & zero corr. linear cryptanalysis (following Sun et al. EC ’16)
- Higher order differential cryptanalysis (following MIDORI/MANTIS)
- And more...
## Cryptanalysis of QARMA – Selected Published Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Time</td>
<td>Data</td>
<td>Memory</td>
</tr>
<tr>
<td>64</td>
<td>4 + 6</td>
<td>N</td>
<td>$2^{116} + 2^{70.1}$</td>
<td>$2^{53}$ CP</td>
<td>$2^{116}$</td>
</tr>
<tr>
<td>64</td>
<td>4 + 4</td>
<td>Y</td>
<td>$2^{33} + 2^{90}$</td>
<td>$2^{16}$ CP</td>
<td>$2^{90}$</td>
</tr>
<tr>
<td>64</td>
<td>4 + 5</td>
<td>Y</td>
<td>$2^{48} + 2^{89}$</td>
<td>$2^{16}$ CP</td>
<td>$2^{89}$</td>
</tr>
<tr>
<td>64</td>
<td>4 + 6</td>
<td>Y</td>
<td>$2^{72}$</td>
<td>$2^{61}$ CP</td>
<td>$2^{78.2}$ bits</td>
</tr>
<tr>
<td>64</td>
<td>4 + 6</td>
<td>Y</td>
<td>$2^{59}$</td>
<td>$2^{59}$ KP</td>
<td>$2^{29.6}$ bits</td>
</tr>
<tr>
<td>64</td>
<td>4 + 7</td>
<td>Y</td>
<td>$2^{120.4}$</td>
<td>$2^{61}$ CP</td>
<td>$2^{116}$</td>
</tr>
<tr>
<td>64</td>
<td>3 + 8</td>
<td>Y</td>
<td>$2^{64.4} + 2^{80}$</td>
<td>$2^{61}$ CP</td>
<td>$2^{61}$</td>
</tr>
<tr>
<td>64</td>
<td>4 + 8</td>
<td>Y</td>
<td>$2^{66.2}$</td>
<td>$2^{48.4}$ CP</td>
<td>$2^{53.70}$</td>
</tr>
<tr>
<td>128</td>
<td>4 + 6</td>
<td>N</td>
<td>$2^{232} + 2^{141.7}$</td>
<td>$2^{105}$ CP</td>
<td>$2^{232}$</td>
</tr>
<tr>
<td>128</td>
<td>5 + 5</td>
<td>Y</td>
<td>$2^{156}$</td>
<td>$2^{88}$ CP</td>
<td>$2^{152}$ bits</td>
</tr>
<tr>
<td>128*</td>
<td>4 + 6</td>
<td>Y</td>
<td>$2^{237.3}$</td>
<td>$2^{122}$ CP</td>
<td>$2^{144}$</td>
</tr>
<tr>
<td>128*</td>
<td>4 + 7</td>
<td>Y</td>
<td>$2^{241.8}$</td>
<td>$2^{122}$ CP</td>
<td>$2^{232}$</td>
</tr>
<tr>
<td>128</td>
<td>4 + 7</td>
<td>Y</td>
<td>$2^{126.1}$</td>
<td>$2^{126.1}$ KP</td>
<td>$2^{71}$ bits</td>
</tr>
</tbody>
</table>
## Implementation results (Power at 10 MHz)

<table>
<thead>
<tr>
<th>Block Cipher</th>
<th>Area (GE)</th>
<th>Power (mW)</th>
<th>Energy (nJ)</th>
<th>Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MIDORI-128</td>
<td>21647</td>
<td>17.60</td>
<td>1.76</td>
<td>18.80</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AES-128</td>
<td>51126</td>
<td>66.33</td>
<td>6.63</td>
<td>25.10</td>
</tr>
<tr>
<td>AES-192</td>
<td>58313</td>
<td>87.47</td>
<td>8.75</td>
<td>28.91</td>
</tr>
<tr>
<td>AES-256</td>
<td>71711</td>
<td>133.74</td>
<td>13.37</td>
<td>33.78</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deoxys-BC-256</td>
<td>61713</td>
<td>108.83</td>
<td>10.88</td>
<td>34.91</td>
</tr>
<tr>
<td>Deoxys-BC-384</td>
<td>74940</td>
<td>145.59</td>
<td>14.56</td>
<td>40.04</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QARMA-128\textsubscript{11}</td>
<td>31242</td>
<td>29.05</td>
<td>2.91</td>
<td>17.87</td>
</tr>
<tr>
<td>QARMA-128\textsubscript{12}</td>
<td>33827</td>
<td>41.59</td>
<td>4.16</td>
<td>19.35</td>
</tr>
<tr>
<td>QARMA-128\textsubscript{13}</td>
<td>36412</td>
<td>48.42</td>
<td>4.84</td>
<td>20.83</td>
</tr>
<tr>
<td>QARMA-128\textsubscript{14}</td>
<td>38998</td>
<td>55.78</td>
<td>5.58</td>
<td>22.32</td>
</tr>
</tbody>
</table>
Memory Encryption
... and many others.
General Purpose Encryption
(Including Data at Rest)
Introducing PANORAmA

PARallelisable
NOOnce
Rotating
Authenticated Encryption
cum
Associated Data
AD Processing

\[ \begin{align*}
A_0 & \quad E_K^2 \| 0 \\
A_1 & \quad E_K^2 \| 1 \\
& \quad \cdots \\
A_{\lambda-2} & \quad E_K^2 \| \lambda-2 \\
pad(A_{\lambda-1}) & \quad E_K^3 \| a \\
\oplus & \quad \lor \\
\oplus & \quad \lor \\
\oplus & \quad \lor \\
\text{Auth} &
\end{align*} \]
Message Processing

\[ \begin{align*}
E_K^{0\|\nu\|0} & \quad \quad E_K^{0\|\nu\|1} \quad \cdots \quad E_K^{0\|\nu\|\ell-2} \\
C_0 & \quad \quad C_1 \quad \cdots \quad C_{\ell-2} \\
\end{align*} \]

\[ \begin{align*}
E_K^{1\|\nu\|m} & \quad \quad E_K^{4\|\nu\|0} \\
C_{\ell-1} & \quad \quad \text{tag} \\
\end{align*} \]

\[ \sum \]

\[ \text{Auth} \]

\[ \text{pad}(M_{\ell-1}) \]

\[ M_0 \quad M_1 \quad M_{\ell-2} \]

\[ \begin{align*}
& 0 \\
\| & \nu \\
\| & 0 \\
K & C_0 \\
\end{align*} \]

\[ \begin{align*}
& 0 \\
\| & \nu \\
\| & 1 \\
K & C_1 \\
\end{align*} \]

\[ \begin{align*}
& 0 \\
\| & \nu \\
\| & \ell-2 \\
K & C_{\ell-2} \\
\end{align*} \]

\[ \begin{align*}
& 1 \\
\| & \nu \\
\| & m \\
K & C_{\ell-1} \\
\end{align*} \]

\[ \begin{align*}
& 4 \\
\| & \nu \\
\| & 0 \\
K & \text{tag} \\
\end{align*} \]
Alternative Message Processing
These schemes can be instantiated with any compatible TBC.

QARMA is suitable.
Nonce Rotation
Entropy is like good Real Estate – always at a Premium

- Suppose you have a TBC with only a 128 bit tweak (or even a 64-bit one!).
- Suppose there is a call for standards that asks for 96-bit nonces and processing of at least $2^{46}$ blocks.
- But in a 128-bit tweak you can only fit, say, 4 bits for tweak domain separation, 96 bits for the nonce, and 28 bits for a block index.
- What can you do?

Design a new cipher with larger tweak?
(Risky and expensive, usually 40+++% more latency, see for instance SKINNY).

If the nonce itself is a counter, increase it every $2^{28}$ blocks – the primitive should return the new nonce value if updated.

In general, change (“rotate”) it every $2^{28}$ blocks – in an unpredictable way, so that nonce, and thus tweak repetitions cannot be exploited.
Entropy is like good Real Estate – always at a Premium

- Suppose you have a TBC with only a 128 bit tweak (or even a 64-bit one!).
- Suppose there is a call for standards that asks for 96-bit nonces and processing of at least $2^{46}$ blocks.
- But in a 128-bit tweak you can only fit, say, 4 bits for tweak domain separation, 96 bits for the nonce, and 28 bits for a block index.
- What can you do?

Design a new cipher with larger tweak? (Risky and expensive, usually 40+++% more latency, see for instance SKINNY).

If the nonce itself is a counter, increase it every $2^{28}$ blocks – the primitive should return the new nonce value if updated.

In general, change (“rotate”) it every $2^{28}$ blocks – in an unpredictable way, so that nonce, and thus tweak repetitions cannot be exploited.
Entropy is like good Real Estate – always at a Premium

- Suppose you have a TBC with only a 128 bit tweak (or even a 64-bit one!).
- Suppose there is a call for standards that asks for 96-bit nonces and processing of at least $2^{46}$ blocks.
- But in a 128-bit tweak you can only fit, say, 4 bits for tweak domain separation, 96 bits for the nonce, and 28 bits for a block index.
- What can you do?

Design a new cipher with larger tweak? (Risky and expensive, usually 40++% more latency, see for instance SKINNY).

If the nonce itself is a counter, increase it every $2^{28}$ blocks – the primitive should return the new nonce value if updated.

In general, change (“rotate”) it every $2^{28}$ blocks – in an unpredictable way, so that nonce, and thus tweak repetitions cannot be exploited.
Entropy is like good Real Estate – always at a Premium

- Suppose you have a TBC with only a 128 bit tweak (or even a 64-bit one!).
- Suppose there is a call for standards that asks for 96-bit nonces and processing of at least $2^{46}$ blocks.
- But in a 128-bit tweak you can only fit, say, 4 bits for tweak domain separation, 96 bits for the nonce, and 28 bits for a block index.
- What can you do?

**Design a new cipher with larger tweak?**
(Risky and expensive, usually 40++% more latency, see for instance SKINNY).

If the nonce itself is a counter, increase it every $2^{28}$ blocks – the primitive should return the new nonce value if updated.

In general, change (“rotate”) it every $2^{28}$ blocks – in an unpredictable way, so that nonce, and thus tweak repetitions cannot be exploited.
Description

1. Start with a nonce $\nu$. Set $N \leftarrow \nu$.

2. Encrypt first $2^{28}$ blocks (a chunk) using tweaks $0000||N||i$, $i$ the block index.
   
   Note that $i$, once put into the tweak, is truncated to the least significant 28 bits.

3. Set
   
   $$N \leftarrow E_{K}^{1111||\nu||(i\gg28)} \text{(magic number)}$$
   
   truncated to 96 bits.

4. Encrypt now up to $2^{28}$ blocks (if finished, stop) using tweaks $1000||N||i$.

5. More to encrypt? go to step 3.
Analysis

1. Eve asks for encryption of all-zero messages, observes repetitions only after

\[ \geq (2 \cdot 2^{28} + 1) \cdot \sqrt{2} \cdot 2^{48} \]

expected blocks.

2. This is roughly $2^{77.5}$. Barely distinguishable from a random function, since collisions may happen anyway after $2^{64.5}$ blocks.

3. (Lower bounds only? $N$ is truncated to 96 $\gg \frac{1}{2}$ 128 bits.)

4. NIST requirements only ask to be able to process at least $2^{46}$ blocks.

5. ???

6. Profit!
Tweak Compression
Entropy is like good Real Estate – always at a Premium

- Suppose you have a TBC with only a 128 bit tweak (or even a 64-bit one!).
- Suppose there is a call for standards that asks for 96-bit nonces and processing of at least $2^{46}$ blocks.
- But in a 128-bit tweak you can only fit, say, 4 bits for tweak domain separation, 96 bits for the nonce, and 28 bits for a block index.
- What can you do?

Design a new cipher with larger tweak?
(Risky and expensive, usually 40++% more latency, see for instance SKINNY).

Nonce rotation? Some may not like it, for whatever reason.

Use a PRF to compress $2^n$ or $3n$ bits worth of tweak material onto just $n$ bits, and modify the mode to make tweak collisions non-exploitable.
Entropy is like good Real Estate – always at a Premium

- Suppose you have a TBC with only a 128 bit tweak (or even a 64-bit one!).
- Suppose there is a call for standards that asks for 96-bit nonces and processing of at least $2^{46}$ blocks.
- But in a 128-bit tweak you can only fit, say, 4 bits for tweak domain separation, 96 bits for the nonce, and 28 bits for a block index.
- What can you do?

Design a new cipher with larger tweak?
(Risky and expensive, usually 40++% more latency, see for instance SKINNY).

Nonce rotation? Some may not like it, for whatever reason.

Use a PRF to compress $2n$ or $3n$ bits worth of tweak material onto just $n$ bits, and modify the mode to make tweak collisions non-exploitable.
Entropy is like good Real Estate – always at a Premium

- Suppose you have a TBC with only a 128 bit tweak (or even a 64-bit one!).
- Suppose there is a call for standards that asks for 96-bit nonces and processing of at least $2^{46}$ blocks.
- But in a 128-bit tweak you can only fit, say, 4 bits for tweak domain separation, 96 bits for the nonce, and 28 bits for a block index.
- What can you do?

Design a new cipher with larger tweak?
(Risky and expensive, usually 40+++% more latency, see for instance SKINNY).

Nonce rotation? Some may not like it, for whatever reason.

Use a PRF to compress $2n$ or $3n$ bits worth of tweak material onto just $n$ bits, and modify the mode to make tweak collisions non-exploitable.
Entropy is like good Real Estate – always at a Premium

- Suppose you have a TBC with only a 128 bit tweak (or even a 64-bit one!).
- Suppose there is a call for standards that asks for 96-bit nonces and processing of at least $2^{46}$ blocks.
- But in a 128-bit tweak you can only fit, say, 4 bits for tweak domain separation, 96 bits for the nonce, and 28 bits for a block index.
- What can you do?

Design a new cipher with larger tweak? (Risky and expensive, usually 40++% more latency, see for instance SKINNY).

Nonce rotation? Some may not like it, for whatever reason.

Use a PRF to compress $2n$ or $3n$ bits worth of tweak material onto just $n$ bits, and modify the mode to make tweak collisions non-exploitable.
How to construct a suitable PRF?

- **Ad hoc?** Nooooooo.
- So let us try putting \( T = T_0 \parallel \cdots \parallel T_{c-1} \) and

\[
\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i) .
\]

- Multiply in Galois fields by keys and add? \( g_i(T_i) = k_i \cdot T_i \) (First idea.) Exploitable. (Hint: collisions give relations between these keys.)
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
  - So let us try putting $T = T_0 \| \cdots \| T_{c-1}$ and
    $$\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i).$$
  - Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.) Exploitable. (Hint: collisions give relations between these keys.)
  - Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting $T = T_0 \parallel \cdots \parallel T_{c-1}$ and
  \[
  \text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i).
  \]
- Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.)
  Exploitable. (Hint: collisions give relations between these keys.)
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting \( T = T_0 \parallel \cdots \parallel T_{c-1} \) and

\[
\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i).
\]

- Multiply in Galois fields by keys and add? \( g_i(T_i) = k_i \cdot T_i \) (First idea.) Exploitable. (Hint: collisions give relations between these keys.)
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting $T = T_0 \parallel \cdots \parallel T_{c-1}$ and

  $$\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i).$$

- Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.) Exploitable. (Hint: collisions give relations between these keys.)
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting $T = T_0 \parallel \cdots \parallel T_{c-1}$ and
  \[
  \text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i) .
  \]
- Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.)
  Exploitable. (Hint: collisions give relations between these keys.)
- Multiply in Galois fields by keys, add, then encrypt?
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting \( T = T_0 \| \cdots \| T_{c-1} \) and

\[
\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i) .
\]

- Multiply in Galois fields by keys and add? \( g_i(T_i) = k_i \cdot T_i \) (First idea.) Exploitable. (Hint: collisions give relations between these keys.)
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting $T = T_0\|\cdots\|T_{c-1}$ and
  
  $\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i)$.

- Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.)
  Exploitable. (Hint: collisions give relations between these keys.)

- Multiply in Galois fields by keys, add, then encrypt?

- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting $T = T_0|| \cdots || T_{c-1}$ and

$$\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i).$$

- Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.) Exploitable. (Hint: collisions give relations between these keys.)
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting $T = T_0 \parallel \cdots \parallel T_{c-1}$ and

  $$\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i) .$$

- Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.) Exploitable. (Hint: collisions give relations between these keys.)
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting $T = T_0 \parallel \cdots \parallel T_{c-1}$ and

  \[
  \text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i) .
  \]

- Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.) Exploitable. (Hint: collisions give relations between these keys.)
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooonoo.
- So let us try putting $T = T_0 \parallel \cdots \parallel T_{c-1}$ and

\[
\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i).
\]

- Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.)
  Exploitable. (Hint: collisions give relations between these keys.)
- Multiply in Galois fields by keys, add, then encrypt?
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

- Ad hoc? Nooooooo.
- So let us try putting $T = T_0 \| \cdots \| T_{c-1}$ and
  
  \[
  \text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i)
  \]

- Multiply in Galois fields by keys and add? $g_i(T_i) = k_i \cdot T_i$ (First idea.) Exploitable. (Hint: collisions give relations between these keys.)
- Construct the PRF by adding PRPs? Much. Better.
How to construct a suitable PRF?

We could try

\[
\text{compressed tweak} = \sum_{i=0}^{c-1} E_K^{\text{tweak}_i}(T_i)
\]

followed by

\[
C = E_K^{\text{compressed tweak}}(P)
\]

This has two problems.

- Kills the advantages of having: A short tweak input; not too many rounds; usage of TBC based modes of operation.
- Collisions give same encryption function.
How to construct a suitable PRF?

We could try

\[
\text{compressed tweak} = \sum_{i=0}^{c-1} E_K^{\text{tweak}_i}(T_i)
\]

followed by

\[
C = E_K^{\text{compressed tweak}}(P).
\]

This has two problems.

- Kills the advantages of having: A short tweak input; not too many rounds; usage of TBC based modes of operation.
- Collisions give same encryption function.
Description

We could try

\[
\text{compressed tweak} = \sum_{i=0}^{c-1} g_i(T_i \oplus K_0)
\]

(I shall call you mini-T, or \(T_{\text{effective}}\))

\[g_i = \text{a few rounds of QARMA with unique round constants, no tweak, key } K_1.\]

For 2 \(n\)-bit long tweaks

\[
P \mapsto E_{K}^{T_{\text{effective}}} (P \oplus o(g_0(T_0 \oplus K_0)))
\]

\[
C \mapsto D_{K}^{T_{\text{effective}}} (C) \oplus o(g_0(T_0 \oplus K_0)).
\]

For 3 \(n\)-bit long tweaks

\[
P \mapsto E_{K}^{T_{\text{effective}}} (P \oplus o(g_0(T_0 \oplus K_0))) \oplus o(g_1(T_1 \oplus K_0))
\]

\[
C \mapsto D_{K}^{T_{\text{effective}}} (C \oplus o(g_1(T_1 \oplus K_0))) \oplus o(g_0(T_0 \oplus K_0)).
\]
Analysis

- Make sure the number of rounds is sufficient to guarantee full diffusion.
- At least 3.
- Use a bit-wise MILP model to ensure that the number of active S-Boxes is 30-ish, resp. 60-ish for each pair of functions \((g_i, g_j)\) in the case of 64, resp. 128-bit tweak ciphers. This makes collisions not only roughly as rare as random, but expensive to construct (the QARMA 4-bit S-Box has \(2^{-2}\) lin/diff biases).
- Also: adding the component functions before/after the encryptions (processed with an orthomorphism) will cause collisions to generate non-identical functions (in fact, translates, which are then not detectable).
- ???
- Profit!!!
A new Hope
A new Hope^H
A new Hope^H^H^H
A new Hope^H^H^H^H
A new Hope^H^H^H^H^H
A new Hope^H^H^H^H^H
Key Schedule
Itai Dinur suggested an Alternating key Construction
QARMA*: Encryption

\[ T \rightarrow h \omega \rightarrow h \omega \rightarrow h \omega \rightarrow \cdots \rightarrow h \omega \rightarrow h \omega \rightarrow h \omega \rightarrow k_0 \rightarrow c_1 \rightarrow k_1 \rightarrow c_2 \rightarrow k_0 \rightarrow c_3 \rightarrow k_1 \rightarrow c_{10} \rightarrow c_{11} \rightarrow M \rightarrow \tau \]

\[ P \rightarrow S \rightarrow \tau M S \rightarrow \tau M S \rightarrow \cdots \rightarrow \tau M S \rightarrow \tau M S \rightarrow \tau M S \rightarrow M \rightarrow \tau \]

\[ C \rightarrow \tilde{S} \rightarrow \tau M \tilde{S} \rightarrow \tau M \tilde{S} \rightarrow \cdots \rightarrow \tau M \tilde{S} \rightarrow \tau M \tilde{S} \rightarrow \tau M \tilde{S} \rightarrow \tau \]

\[ \tilde{h} \omega \rightarrow \tilde{h} \omega \rightarrow \tilde{h} \omega \rightarrow \tilde{h} \omega \rightarrow \cdots \rightarrow \tilde{h} \omega \rightarrow \tilde{h} \omega \rightarrow \tilde{h} \omega \rightarrow \tau \]

\[ k_0 + O(k_1) \]

\[ M \cdot k_0 + o(k_1) \]
QARMA*: Decryption
How to use a Characteristic/Distinguisher

The map $x \mapsto o(x)$ is linear, bijective, and also $x \mapsto x + o(x)$ is bijective. This makes in general $\Delta'$ useless unless $k_0$ is guessed.
More Remarks

Over characteristic 2 algebras, if $o(\cdot)$ is an orthomorphism, then $o^2(\cdot)$ #metoo.

Proof: First, note that $o^2(\cdot)$ is clearly a bijective linear map. Then $x \mapsto y := x + o(x) \mapsto y + o(y) = x + o(x) + o(x) + o^2(x) = x + o^2(x)$ as a composition of bijections is a bijection. □

Hence, $o^4(\cdot)$, $o^8(\cdot)$, $o^{2^i}(\cdot)$ are orthomorphisms as well.

Idea: (Perchance, to dream?) Use a simple key schedule where we alternate between partial keys, and modify values using functions that, as much as possible, are pairwise “mutually orthomorphic.”

(Ay, there's the rub. Doing it elegantly. Also $o(\cdot)$ is one XOR, then $o^{2^i}(\cdot)$ is $2^i$ XORs.)

(The reference is to Shakespeare, not Bochum.)
More Remarks

Over characteristic 2 algebras, if \( o(\cdot) \) is an orthomorphism, then \( o^2(\cdot) \) #metoo.

Proof: First, note that \( o^2(\cdot) \) is clearly a bijective linear map. Then \( x \mapsto y := x + o(x) \mapsto y + o(y) = x + o(x) + o(x) + o^2(x) = x + o^2(x) \) as a composition of bijections is a bijection. □

Hence, \( o^4(\cdot), o^8(\cdot), o^{2^i}(\cdot) \) are orthomorphisms as well.

Idea: (Perchance, to dream?) Use a simple key schedule where we alternate between partial keys, and modify values using functions that, as much as possible, are pairwise “mutually orthomorphic.”

(Ay, there’s the rub. Doing it elegantly. Also \( o(\cdot) \) is one XOR, then \( o^{2^i}(\cdot) \) is \( 2^i \) XORs.)

(The reference is to Shakespeare, not Bochum.)
More Remarks

Over characteristic 2 algebras, if $o(\cdot)$ is an orthomorphism, then $o^2(\cdot) \#\text{metoo}.$

**Proof:** First, note that $o^2(\cdot)$ is clearly a bijective linear map. Then $x \mapsto y := x + o(x) \mapsto y + o(y) = x + o(x) + o(x) + o^2(x) = x + o^2(x)$ as a composition of bijections is a bijection. \(\square\)

Hence, $o^4(\cdot), o^8(\cdot), o^{2^i}(\cdot)$ are orthomorphisms as well.

**Idea:** (Perchance, to dream?) Use a simple key schedule where we alternate between partial keys, and modify values using functions that, as much as possible, are pairwise “mutually orthomorphic.”

(Ay, there's the rub. Doing it elegantly. Also $o(\cdot)$ is one XOR, then $o^{2^i}(\cdot)$ is $2^i$ XORs.)

(The reference is to Shakespeare, not Bochum.)
More Remarks

Over characteristic 2 algebras, if \( o(\cdot) \) is an orthomorphism, then \( o^2(\cdot) \) #metoo.

**Proof**: First, note that \( o^2(\cdot) \) is clearly a bijective linear map.

Then \( x \mapsto y := x + o(x) \mapsto y + o(y) = x + o(x) + o(x) + o^2(x) = x + o^2(x) \) as a composition of bijections is a bijection. \( \square \)

Hence, \( o^4(\cdot), o^8(\cdot), o^{2^i}(\cdot) \) are orthomorphisms as well.

**Idea**: (Perchance, to dream?) Use a simple key schedule where we alternate between partial keys, and modify values using functions that, as much as possible, are pairwise “mutually orthomorphic.”

(Ay, there’s the *rub*. Doing it elegantly. Also \( o(\cdot) \) is one XOR, then \( o^{2^i}(\cdot) \) is \( 2^i \) XORs.)

(The reference is to Shakespeare, not Bochum.)
More Remarks

Over characteristic 2 algebras, if $o(\cdot)$ is an orthomorphism, then $o^2(\cdot)$ #metoo.

**Proof:** First, note that $o^2(\cdot)$ is clearly a bijective linear map.
Then $x \leftrightarrow y := x + o(x) \leftrightarrow y + o(y) = x + o(x) + o(x) + o^2(x) = x + o^2(x)$ as a composition of bijections is a bijection. □

Hence, $o^4(\cdot)$, $o^8(\cdot)$, $o^{2^i}(\cdot)$ are orthomorphisms as well.

**Idea:** (Perchance, to dream?) Use a simple key schedule where we alternate between partial keys, and modify values using functions that, as much as possible, are pairwise “mutually orthomorphic.”

(Ay, there’s the rub. Doing it elegantly. Also $o(\cdot)$ is one XOR, then $o^{2^i}(\cdot)$ is $2^i$ XORs.)

(The reference is to Shakespeare, not Bochum.)
This is a possible starting point. To do: reduce (partial) sliding; and how to properly construt the forward and backward paths of a reflection cipher.

Nearly all of the cryptanalysis done so far would apply almost verbatim or have higher complexities. Invariant Subspace analysis would have to be re-run.

If you find a good way, please do it with us and maybe also Yannick Seurin. Really, be cooperative. Orr's daughter knows how to use an AK47.
This is a possible starting point. To do: reduce (partial) sliding; and how to properly construct the forward and backward paths of a reflection cipher.

Nearly all of the cryptanalysis done so far would apply almost verbatim or have higher complexities. Invariant Subspace analysis would have to be re-run.

If you find a good way, please do it with us and maybe also Yannick Seurin. Really, be cooperative. Orr’s daughter knows how to use an AK47.
This is a possible starting point. To do: reduce (partial) sliding; and how to properly construct the forward and backward paths of a reflection cipher.

Nearly all of the cryptanalysis done so far would apply almost verbatim or have higher complexities. Invariant Subspace analysis would have to be re-run.

If you find a good way, please do it with us and maybe also Yannick Seurin. Really, be cooperative. Orr's daughter knows how to use an AK47.
This is a possible starting point. To do: reduce (partial) sliding; and how to properly construct the forward and backward paths of a reflection cipher.

Nearly all of the cryptanalysis done so far would apply almost verbatim or have higher complexities. Invariant Subspace analysis would have to be re-run.

If you find a good way, please do it with us and maybe also Yannick Seurin.

Really, be cooperative. Orr's daughter knows how to use an AK47.
This is a possible starting point. To do: reduce (partial) sliding; and how to properly construct the forward and backward paths of a reflection cipher.

Nearly all of the cryptanalysis done so far would apply almost verbatim or have higher complexities. Invariant Subspace analysis would have to be re-run.

If you find a good way, please do it with us and maybe also Yannick Seurin.

Really, be cooperative. Orr’s daughter knows how to use an AK47.
This is a possible starting point. To do: reduce (partial) sliding; and how to properly construct the forward and backward paths of a reflection cipher.

Nearly all of the cryptanalysis done so far would apply almost verbatim or have higher complexities. Invariant Subspace analysis would have to be re-run.

If you find a good way, please do it with us and maybe also Yannick Seurin. Really, be cooperative. Orr’s daughter knows how to use an AK47.
What about Qameleon?
Introducing Qameleon

QARMA plus
Authentication for
MEmories that
Let
ExfiltratìON
Qameleon

- **Qameleon** is an AEAD suite of ciphers for various applications.
- In a nutshell, it is PANORAmA instantiated with QARMA, using in some cases Nonce Rotation or Tweak Compression to be able to cover all requirements.
- For those that have read the NIST LWC competition call, they asked for something essentially already proven and well analysed. They stopped short of saying “just give us ASCON and we shall pretend to have a look at the rest.” The next thing may just be a known and studied TBC with an OCB-like mode.
- We use QARMA and a subset of ΘCB (plus NR & TC that come with proofs).
- In our opinion these constructions are even better understood than sponges. ISO folks in this room, take note!
**Qameleon**

- **Qameleon** is an AEAD suite of ciphers for various applications.

- In a nutshell, it is **PANORAmA** instantiated with **QARMA**, using in some cases **Nonce Rotation** or **Tweak Compression** to be able to cover all requirements.

- For those that have read the NIST LWC competition call, they asked for something essentially already proven and well analysed. They stopped short of saying “just give us **ASCON** and we shall pretend to have a look at the rest.” The next thing may just be a known and studied TBC with an OCB-like mode.

- We use **QARMA** and a subset of **$\Theta$CB** (plus NR & TC that come with proofs).

- In our opinion these constructions are even better understood than sponges. **ISO folks in this room, take note!**
Qameleon

- Qameleon is an AEAD suite of ciphers for various applications.

- In a nutshell, it is PANORAmA instantiated with QARMA, using in some cases Nonce Rotation or Tweak Compression to be able to cover all requirements.

- For those that have read the NIST LWC competition call, they asked for something essentially already proven and well analysed. They stopped short of saying “just give us ASCON and we shall pretend to have a look at the rest.” The next thing may just be a known and studied TBC with an OCB-like mode.

- We use QARMA and a subset of ΘCB (plus NR & TC that come with proofs).

- In our opinion these constructions are even better understood than sponges. ISO folks in this room, take note!
Qameleon

- Qameleon is an AEAD suite of ciphers for various applications.

- In a nutshell, it is PANORAmA instantiated with QARMA, using in some cases Nonce Rotation or Tweak Compression to be able to cover all requirements.

- For those that have read the NIST LWC competition call, they asked for something essentially already proven and well analysed. They stopped short of saying “just give us ASCON and we shall pretend to have a look at the rest.” The next thing may just be a known and studied TBC with an OCB-like mode.

- We use QARMA and a subset of ΘCB (plus NR & TC that come with proofs).

- In our opinion these constructions are even better understood than sponges. ISO folks in this room, take note!
Qameleon

- Qameleon is an AEAD suite of ciphers for various applications.

- In a nutshell, it is PANORAmA instantiated with QARMA, using in some cases Nonce Rotation or Tweak Compression to be able to cover all requirements.

- For those that have read the NIST LWC competition call, they asked for something essentially already proven and well analysed. They stopped short of saying “just give us ASCON and we shall pretend to have a look at the rest.” The next thing may just be a known and studied TBC with an OCB-like mode.

- We use QARMA and a subset of ΘCB (plus NR & TC that come with proofs).

- In our opinion these constructions are even better understood than sponges. ISO folks in this room, take note!
## Implementation results (Power at 10 MHz)

<table>
<thead>
<tr>
<th>Variant</th>
<th>Block Cipher</th>
<th>Optimization</th>
<th>Area (GE)</th>
<th>Power(mW)</th>
<th>Delay(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) qameleon12812896gpv1</td>
<td>QARMA-128\textsubscript{11}</td>
<td>Area</td>
<td>35053</td>
<td>35.0</td>
<td>20.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed</td>
<td>59678</td>
<td>69.4</td>
<td>9.92</td>
</tr>
<tr>
<td></td>
<td>QARMA-128\textsubscript{14}</td>
<td>Area</td>
<td>42811</td>
<td>53.9</td>
<td>25.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed</td>
<td>74177</td>
<td>109.8</td>
<td>11.91</td>
</tr>
<tr>
<td>(B) qameleon128128128tcgpv1</td>
<td>QARMA-128\textsubscript{11}</td>
<td>Area</td>
<td>46018</td>
<td>45.8</td>
<td>23.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed</td>
<td>90138</td>
<td>100.3</td>
<td>10.42</td>
</tr>
<tr>
<td>(C) qameleon12812864gpv1</td>
<td>QARMA-128\textsubscript{14}</td>
<td>Area</td>
<td>42379</td>
<td>52.9</td>
<td>24.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed</td>
<td>96708</td>
<td>86.7</td>
<td>9.90</td>
</tr>
<tr>
<td>(D) qameleon1286464mev1</td>
<td>QARMA-128\textsubscript{11}</td>
<td>Area</td>
<td>32457</td>
<td>28.6</td>
<td>18.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed</td>
<td>56873</td>
<td>61.9</td>
<td>9.00</td>
</tr>
<tr>
<td>(E) qameleon6464tcmev1</td>
<td>QARMA-64\textsubscript{7}(TC)</td>
<td>Area</td>
<td>15702</td>
<td>9.5</td>
<td>16.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed</td>
<td>19408</td>
<td>16.2</td>
<td>7.90</td>
</tr>
<tr>
<td>(F) qameleon6464nnmev1</td>
<td>QARMA-64\textsubscript{9}</td>
<td>Area</td>
<td>13779</td>
<td>9.8</td>
<td>16.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed</td>
<td>22892</td>
<td>19.6</td>
<td>8.00</td>
</tr>
<tr>
<td>ΘCB</td>
<td>Deoxys-BC-384</td>
<td>Area</td>
<td>77967</td>
<td>112.1</td>
<td>57.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed</td>
<td>99128</td>
<td>178.7</td>
<td>29.91</td>
</tr>
</tbody>
</table>
The QARMA Chakra
Questions?