QUARK
a lightweight hash

Jean-Philippe Aumasson

with Luca Henzen, Willi Meier, María Naya-Plasencia
HASH, x. There is no definition for this word—nobody knows what hash is.
A. Bierce, The Devil’s Dictionary
HASH, x. There is no definition for this word—nobody knows what hash is.
A. Bierce, The Devil’s Dictionary

Arbitrary long string $\xrightarrow{HASH}$ Short random-looking string

Most common hashes: MD5 (128 bits), SHA-1 (160 bits)
Hashing in dedicated IC’s (as RFID tags’ chips)

- Identification protocols
- Message authentication

MD5 and SHA-1 generally too big (6000+ GE)

Smallest known proposal: PRESENT-based hashes by Bogdanov et al. (CHES 2008)

- 64-bit hash: 1600 GE
- 128-bit hash: 2330 GE
Hashing in dedicated IC’s (as RFID tags’ chips)

- Identification protocols
- Message authentication

MD5 and SHA-1 generally too big (6000+ GE)

Smallest known proposal: PRESENT-based hashes by Bogdanov et al. (CHES 2008)
  - 64-bit hash: 1600 GE
  - 128-bit hash: 2330 GE
QUARK

- 128-bit hash: 1379 GE
- 160-bit hash: 1702 GE
- 224-bit hash: 2296 GE
Let the design define security, not the hash length

Folklore: security defined by the hash length
⇒ restricts the diversity of designs
Let the design define security, not the hash length

Folklore: security defined by the hash length
⇒ restricts the diversity of designs

Proposal by Bertoni/Daemen/Peeters/Van Assche: security defined by a parameter (the capacity)
Sponge functions

1. “Absorb” chunks of message $m_0, m_1, \ldots$
2. “Squeeze” to extract hash value $z_0, z_1, \ldots$
Sponge reduce memory requirements, if one tolerates suboptimal 2nd preimage resistance

Example:

- hash length $n = 128$
- capacity $c = 128$
- block size $r = 8$

If $P$ is secure, guaranteed security of

- 128 bits against preimages (cf. next talk at 9h50)
- 64 bits against 2nd preimages
- 64 bits against collisions
Sponge reduce memory requirements, if one tolerates suboptimal 2nd preimage resistance

Example:

- hash length $n = 128$
- capacity $c = 128$
- block size $r = 8$

If $P$ is secure, guaranteed security of

- 128 bits against preimages (cf. next talk at 9h50)
- 64 bits against 2nd preimages
- 64 bits against collisions

$\Rightarrow$ storage of 136 bits, instead of 256 with traditional constructions (save $\approx 1000$ GE)
The 3 QUARK flavors

<table>
<thead>
<tr>
<th>Flavor</th>
<th>Capacity</th>
<th>Block</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-QUARK</td>
<td>128</td>
<td>8</td>
<td>136</td>
</tr>
<tr>
<td>D-QUARK</td>
<td>160</td>
<td>16</td>
<td>176</td>
</tr>
<tr>
<td>T-QUARK</td>
<td>224</td>
<td>32</td>
<td>256</td>
</tr>
</tbody>
</table>
Don’t reinvent the wheel: borrow from the best lightweight algorithms

Grain (stream cipher)
Hell/Johansson/Meier, ECRYPT eSTREAM portfolio

KATAN (block cipher)
De Cannière/Dunkelman/Knežević, CHES 2009
Grain

What we borrow

- Update mechanism with 3 boolean functions
- High-degree boolean functions for rapid growth of internal nonlinearity
- Internal parallelism to allow space/time implementation trade-offs
What we borrow

- 2 NFSR’s rather than 1 NFSR and 1 LFSR for better nonlinearity, and to avoid too much dissymmetry
- Auxiliary LFSR as a counter and breaking round self-similarity
QUARK’s $P$ permutation

- Load input in $(X, Y)$ and a constant in $L$
- Clock $4 \times (|X| + |Y|)$ times
- Return final value of $(X, Y)$
Confidence in security based on

- Well-chosen boolean functions and taps
- High number of rounds ($4 \times$ that of Grain)
- Benchmarks with basic cube and differential attacks
  \[ \Rightarrow 22\% \text{ of the rounds broken (nonrandom), consistent for all 3 flavors of QUARK} \]

<table>
<thead>
<tr>
<th></th>
<th>Total rounds</th>
<th>Rounds attacked in $2^8$</th>
<th>in $2^{16}$</th>
<th>in $2^{24}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-QUARK</td>
<td>544</td>
<td>109</td>
<td>111</td>
<td>114</td>
</tr>
<tr>
<td>D-QUARK</td>
<td>704</td>
<td>144</td>
<td>144</td>
<td>148</td>
</tr>
<tr>
<td>T-QUARK</td>
<td>1024</td>
<td>213</td>
<td>220</td>
<td>222</td>
</tr>
</tbody>
</table>

Sponge proof \[\Rightarrow \text{any attack must exploit a flaw in } P\]
VHDL implementation

```vhdl
signal QuarkStatexFP, QuarkStatexFD : std_logic_vector(0 to WWIDTH*8-1);
signal PermOutxF : std_logic_vector(0 to WWIDTH*8-1);
signal X, Y : std_logic;
signal LxD, LxD, LoutxD : std_logic_vector(0 to LWIDTH*8/2-1);
signal Xn, Yn, Ln, h, Xnn, Ynn : std_logic;
signal OUTENXE, FreezeEX : std_logic;
signal DINxD : std_logic_vector(0 to RWF*8-1);
signal IR, IW, ID : std_logic_vector(7 downto 0);
```

2 architectures of each flavor

- Serial (1 instance of each boolean function)
- Parallel (8 or 16 instances of each function)

Tech: UMC 0.18 µm 1P6M CMOS

Place-and-route, and simulations at 100 kHz

Goal: minimize area and power consumption
parallelization degree

permutation

internal state

NFSR X

NFSR Y

LFSR L

f

g

h

parallelization degree
### QUARK vs. PRESENT-based hashes (serial)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DM-PRESENT-80</td>
<td>64</td>
<td>64</td>
<td>32</td>
<td>1600</td>
<td>14.63</td>
<td>1.83</td>
<td>-</td>
</tr>
<tr>
<td>DM-PRESENT-128</td>
<td>64</td>
<td>64</td>
<td>32</td>
<td>1886</td>
<td>22.90</td>
<td>2.94</td>
<td>-</td>
</tr>
<tr>
<td>H-PRESENT-128</td>
<td>128</td>
<td>128</td>
<td>64</td>
<td>2330</td>
<td>11.45</td>
<td>6.44</td>
<td>-</td>
</tr>
<tr>
<td>U-QUARK</td>
<td>128</td>
<td>64</td>
<td>64</td>
<td>1379</td>
<td>1.47</td>
<td>2.44</td>
<td>2.96</td>
</tr>
<tr>
<td>D-QUARK</td>
<td>160</td>
<td>80</td>
<td>80</td>
<td>1702</td>
<td>2.27</td>
<td>3.10</td>
<td>3.95</td>
</tr>
<tr>
<td>T-QUARK</td>
<td>224</td>
<td>112</td>
<td>112</td>
<td>2296</td>
<td>3.13</td>
<td>4.35</td>
<td>5.53</td>
</tr>
</tbody>
</table>
### QUARK vs. PRESENT-based hashes (parallel)

<table>
<thead>
<tr>
<th></th>
<th>Security</th>
<th>Area [GE]</th>
<th>Thr. [kbps]</th>
<th>Power [µW]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre</td>
<td>2nd</td>
<td>Col</td>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td>DM-PRESENT-80</td>
<td>64</td>
<td>64</td>
<td>32</td>
<td>2213</td>
<td>242.42</td>
</tr>
<tr>
<td>DM-PRESENT-128</td>
<td>64</td>
<td>64</td>
<td>32</td>
<td>2530</td>
<td>387.88</td>
</tr>
<tr>
<td>H-PRESENT-128</td>
<td>128</td>
<td>128</td>
<td>64</td>
<td>4256</td>
<td>200.00</td>
</tr>
<tr>
<td>U-QUARK×8</td>
<td>128</td>
<td>64</td>
<td>64</td>
<td>2392</td>
<td>11.76</td>
</tr>
<tr>
<td>D-QUARK×8</td>
<td>160</td>
<td>80</td>
<td>80</td>
<td>2819</td>
<td>18.18</td>
</tr>
<tr>
<td>T-QUARK×16</td>
<td>224</td>
<td>112</td>
<td>112</td>
<td>4640</td>
<td>50.00</td>
</tr>
</tbody>
</table>
QUARK is indeed lightweight!

- 128-bit preimage resistance with only 1379 GE
- 224-bit preimage resistance with only 2296 GE
- Low power consumption (<5 µW)

Compared to PRESENT-based hashes

- Better security/area ratio
- Lower throughput (and thus more energy/bit)
- Better security margin
  ($\approx 80\%$ of PRESENT’s rounds attacked)
A multi-purpose primitive!

Like most hash functions, QUARK can be used as

- PRF
- MAC
- PRNG
- stream cipher
- entropy extractor
- parallel tree-hash
- key derivation function
- etc.
A hash is always secure, until it’s broken.
(adapted from) Y. Berra

Please cryptanalyze QUARK!

Full version of the paper, VHDL and C code available at

http://131002.net/quark/
QUARK
a lightweight hash

Jean-Philippe Aumasson

with Luca Henzen, Willi Meier, María Naya-Plasencia