QUARK
a lightweight hash

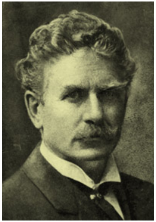
Jean-Philippe Aumasson

NAGRAVISION
KUDELSKI GROUP

with Luca Henzen, Willi Meier, María Naya-Plasencia

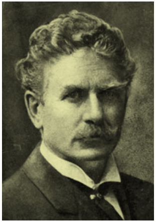
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

n|w University of Applied Sciences Northwestern Switzerland
School of Engineering



HASH, x. There is no definition for this word—nobody knows what hash is.

A. Bierce, The Devil's Dictionary



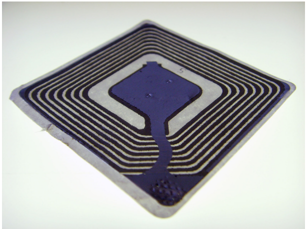
HASH, x. There is no definition for this word—nobody knows what hash is.

A. Bierce, The Devil's Dictionary

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

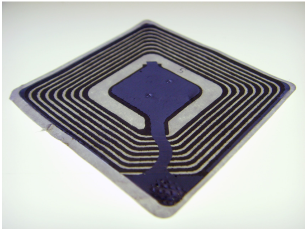
Most common hashes: MD5 (128 bits), SHA-1 (160 bits)

Hashing in dedicated IC's (as RFID tags' chips)



- Identification protocols
- Message authentication

Hashing in dedicated IC's (as RFID tags' chips)



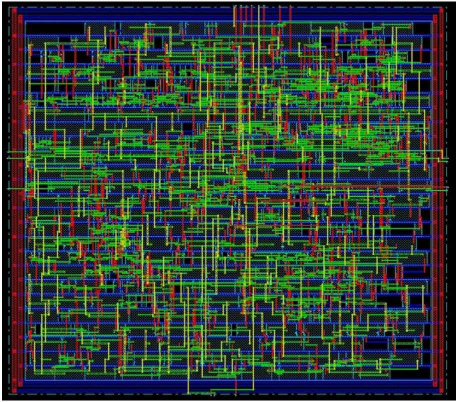
- Identification protocols
- Message authentication

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
$\Rightarrow$ restricts the diversity of designs

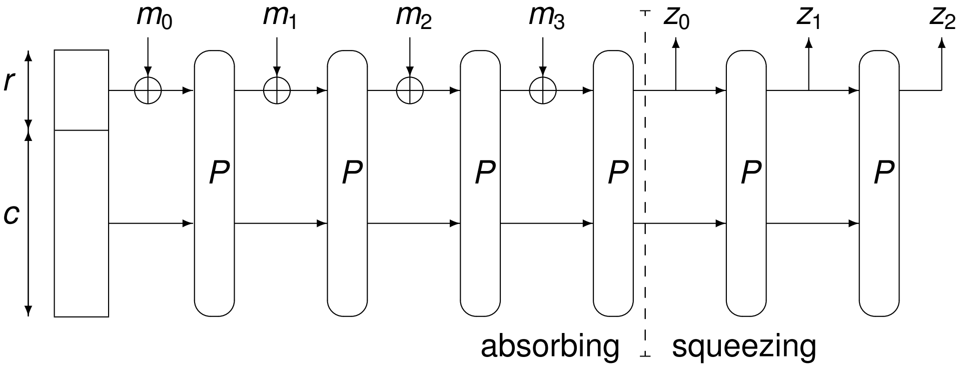
Let the design define security, not the hash length

Folklore: security defined by the hash length
$\Rightarrow$ 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:

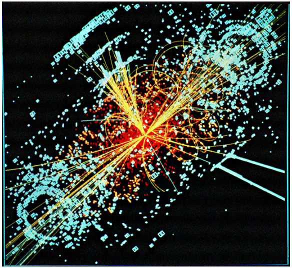
- hash length $n = 128$
- capacity $c = 128$
- block size $r = 8$

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



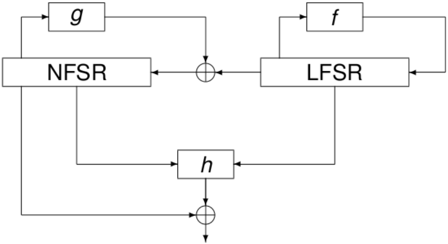
	capacity $n = c$	block r	state $c + r$
U-QUARK	128	8	136
D-QUARK	160	16	176
T-QUARK	224	32	256

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

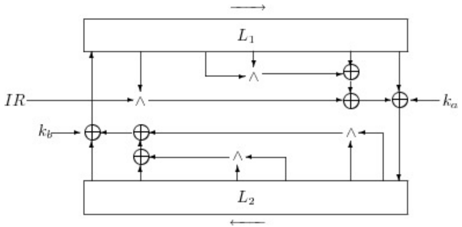
Grain



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

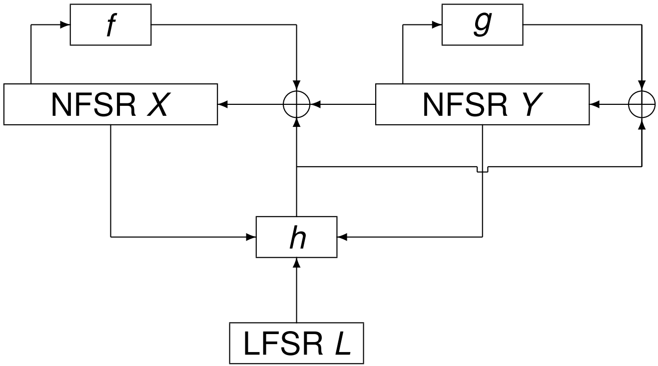
KATAN



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% of the rounds broken (nonrandom), consistent for all 3 flavors of QUARK

	Total	Rounds attacked		
	rounds	in 2^8	in 2^{16}	in 2^{24}
U-QUARK	544	109	111	114
D-QUARK	704	144	144	148
T-QUARK	1024	213	220	222

Sponge proof $\Rightarrow$ any attack must exploit a flaw in P

VHDL implementation

```
signal QuarkStatexDP, QuarkStatexDN : std_logic_vector(0 to WWIDTH*8-1);
signal PermOutxD                    : std_logic_vector(0 to WWIDTH*8-1);
signal X, Y                         : std_logic_vector(0 to WWIDTH*8/2-1);
signal LxDN, LxDP, LoutxD           : std_logic_vector(0 to LWIDTH-1);
signal Xn, Yn, Ln, h, Xnn, Ynn      : std_logic;
signal OUTENxE, FreezexS            : std_logic;
signal DInxD                        : std_logic_vector(0 to RATE*8-1);
signal IR, IW, ID                   : std_logic_vector(7 downto 0);
```

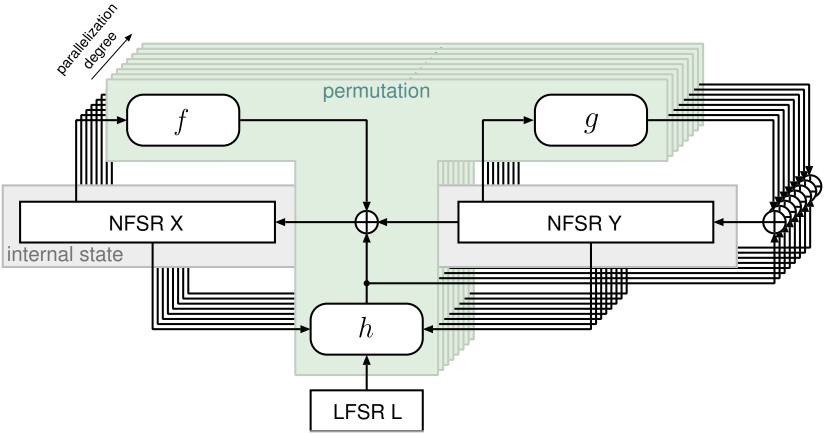
2 architectures of each flavor

- 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

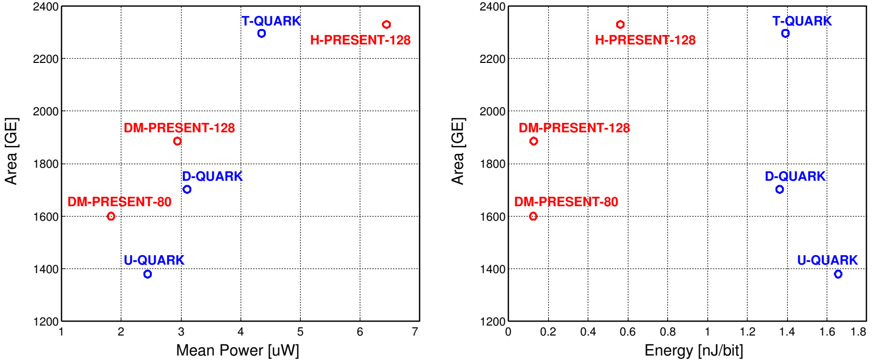

parallelization degree
permutation
f
g
NFSR X
NFSR Y
internal state
h
LFSR L

QUARK vs. PRESENT-based hashes (serial)

	Security			Area	Thr.	Power [µW]	
	Pre	2nd	Col	[GE]	[kbps]	Mean	Peak
DM-PRESENT-80	64	64	32	1600	14.63	1.83	-
DM-PRESENT-128	64	64	32	1886	22.90	2.94	-
H-PRESENT-128	128	128	64	2330	11.45	6.44	-
U-QUARK	128	64	64	1379	1.47	2.44	2.96
D-QUARK	160	80	80	1702	2.27	3.10	3.95
T-QUARK	224	112	112	2296	3.13	4.35	5.53

QUARK vs. PRESENT-based hashes (parallel)

	Security			Area	Thr.	Power [µW]	
	Pre	2nd	Col	[GE]	[kbps]	Mean	Peak
DM-PRESENT-80	64	64	32	2213	242.42	6.28	-
DM-PRESENT-128	64	64	32	2530	387.88	7.49	-
H-PRESENT-128	128	128	64	4256	200.00	8.09	-
U-QUARK×8	128	64	64	2392	11.76	4.07	4.84
D-QUARK×8	160	80	80	2819	18.18	4.76	5.80
T-QUARK×16	224	112	112	4640	50.00	8.39	9.79


2400
2200
2000
1800
1600
1400
1200
Area [GE]
1
2
3
4
5
6
7
Mean Power [uW]
T-QUARK
H-PRESENT-128
DM-PRESENT-128
D-QUARK
DM-PRESENT-80
U-QUARK
2400
2200
2000
1800
1600
1400
1200
Area [GE]
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
Energy [nJ/bit]
H-PRESENT-128
T-QUARK
DM-PRESENT-128
D-QUARK
DM-PRESENT-80
U-QUARK

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

NAGRAVISION
KUDELSKI GROUP

with Luca Henzen, Willi Meier, María Naya-Plasencia

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

n|w
University of Applied Sciences Northwestern Switzerland
School of Engineering