Keccak

An update

Guido Bertoni[1] Joan Daemen[1]
Michaël Peeters[2] Gilles Van Assche[1]

[1]STMicroelectronics

[2]NXP Semiconductors

Third SHA-3 candidate conference, Washington DC
March 22-23, 2012

Outline

1 KECCAK uses a wide permutation

2 KECCAK's safety margins

3 KECCAK's cryptanalysis strengths

4 KECCAK's offering

5 Conclusions

Outline

1 KECCAK uses a wide permutation

2 KECCAK's safety margins

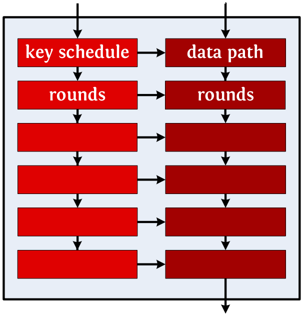
3 KECCAK's cryptanalysis strengths

4 KECCAK's offering

5 Conclusions

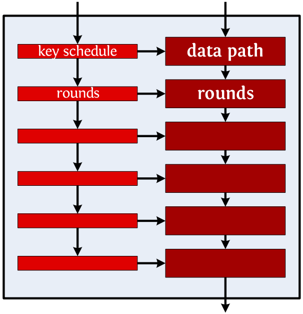
Block cipher versus permutation

- No diffusion from data path to key (and tweak) schedule
  - local collisions
- Sometimes **lightweight** key schedule
- Let's remove these artificial barriers…
- That's a permutation!



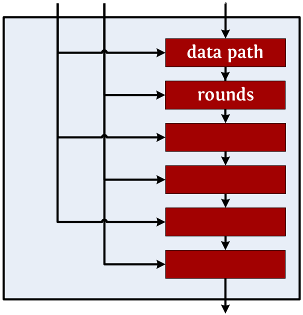
Block cipher versus permutation

- No diffusion from data path to key (and tweak) schedule
  - local collisions
- Sometimes **lightweight** key schedule
- Let's remove these artificial barriers...
- That's a permutation!



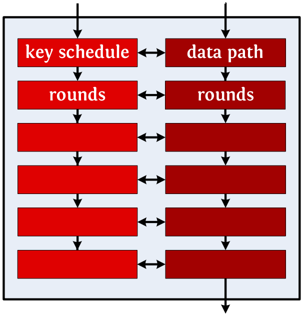
Block cipher versus permutation

- No diffusion from data path to key (and tweak) schedule
  - local collisions
- Sometimes **lightweight** key schedule
- Let's remove these artificial barriers...
- That's a permutation!



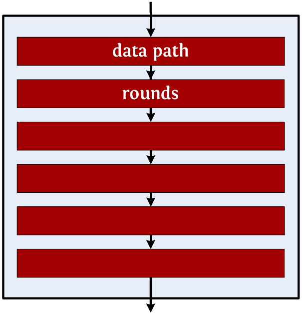
Block cipher versus permutation

- No diffusion from data path to key (and tweak) schedule
  - local collisions
- Sometimes lightweight key schedule
- Let's remove these artificial barriers...
- That's a permutation!



Block cipher versus permutation

- No diffusion from data path to key (and tweak) schedule
  - local collisions
- Sometimes **lightweight** key schedule
- Let's remove these artificial barriers...
- That's a permutation!



But it makes Keccak big!?!

Myth: Keccak requires a lot of working memory

A 1600-bit wide permutation indeed!

Fact: Keccak fits in less than 280 bytes of RAM

- Keccak is among the most compact [XBX]
  - On ARM: fastest = least RAM
- No additional storage required for message or feedforward
- Lightweight hash function proposals are all sponges!
  [Quark, Photon, Spongent]

Data path width of SHA-3 finalists

	D	RAM	Comments
Blake-256	512	1920	lightweight
Blake-512	1024	3904	key schedule
Grøstl-256	512	2048	two permutations
Grøstl-512	1024	5088	in parallel
JH	1024	2624	
Keccak	1600	1856	
Skein	512	2888	lightweight key schedule

RAM usage (in bits) from [XBX]+[Feichtner], min. across platforms

An aside: zero-sum distinguishers

	Zero-sum set size	Exploited property
Blake-256's rounds	2^{512}	keyed permutation
Blake-512's rounds	2^{1024}	keyed permutation
Grøstl's P or Q_{512}	2^{509}	non-maximal degree in the middle
Grøstl's P or Q_{1024}	2^{1024}	permutation
JH's E_8	2^{1024}	permutation
Keccak-f[1600]	2^{1575}	non-maximal degree in the middle
Threefish-512	2^{512}	keyed permutation

distinguisher on Keccak-f[1600]

... yet ...

largest size among finalists

Outline

1 Keccak uses a wide permutation

2 Keccak's safety margins

3 Keccak's cryptanalysis strengths

4 Keccak's offering

5 Conclusions

A distinguisher for KECCAK-f breaks KECCAK?

Myth: KECCAK needs the permutation to admit no distinguisher

Consequence of expressing the **hermetic sponge strategy**

- **No** distinguisher on KECCAK-f!

Flat sponge claim on KECCAK

Fact: Hermetic strategy provides safety margin w.r.t. flat claim

A distinguisher for Keccak-f breaks Keccak?

Myth: Keccak needs the permutation to admit no distinguisher

Flat sponge claim on Keccak

No attack below complexity $2^{c/2}$ (if not easier on random oracle)

- Covers **all attacks**, not only (second) preimage and collision

Fact: Hermetic strategy provides safety margin w.r.t. flat claim

A distinguisher for Keccak-*f* breaks Keccak?

Myth: Keccak needs the permutation to admit no distinguisher

Flat sponge claim on Keccak

Fact: Hermetic strategy provides safety margin w.r.t. flat claim

- No distinguisher on Keccak-*f*, except for zero-sums (2^{1575})
  - Hermetic for DC/LC, symmetries, constrained I/O, etc.
- To invalidate claim, the distinguisher on Keccak-*f* must be:
  - applicable on the Keccak sponge function
  - $< 2^{800}$ for any instance of Keccak
  - $< 2^n$ for any n-bit SHA-3 candidate

Safety margin in the choice of capacity

Flat sponge claim on KECCAK

No attack below complexity $2^{c/2}$ (if not easier on random oracle)

- Covers **all attacks**, not only (second) preimage and collision

"KECCAK-256" = KECCAK$[c = 512]$

- Can output 512 bits and provide 2^{256} collision resistance
- Is sufficient for all security strength levels of [NIST SP 800-57]
  - Blake-512 and Grøstl-512 needed for generic 256-bit security [Andreeva, Mennink, Preneel, Škrobot]

"KECCAK-512" = KECCAK$[c = 1024]$

- Could output 1024 bits and provide 2^{512} collision resistance
- Only if 2^{512} (second) preimage resistance is wanted

Safety margin in the number of rounds

- Keccak-f has 24 rounds
- Sufficient #rounds for security claim on Keccak: 13 rounds
  Estimation from [Keccak reference]

Currently known results
keep us confident about this estimation

What if performance is scaled to security margin?

Safety margin in the number of rounds

- KECCAK-*f* has 24 rounds
- Sufficient #rounds for security claim on KECCAK: 13 rounds
  Estimation from [KECCAK reference]

Currently known results
keep us confident about this estimation

What if performance is scaled to security margin?

Outline

1 KECCAK uses a wide permutation

2 KECCAK's safety margins

3 KECCAK's cryptanalysis strengths

4 KECCAK's offering

5 Conclusions

Third-party cryptanalysis of Keccak

Distinguishers on Keccak-$f[1600]$

Rounds	Work	
3	low	CICO problem [Aumasson, Khovratovich, 2009]
4	low	cube testers [Aumasson, Khovratovich, 2009]
8	2^{491}	unaligned rebound [Duc, Guo, Peyrin, Wei, FSE 2012]
24	2^{1574}	zero-sum [Duan, Lai, ePrint 2011] [Boura, Canteaut, De Cannière, FSE 2011]

Academic-complexity attacks on Keccak

- 6-8 rounds: second preimage [Bernstein, 2010]
  - *slightly faster* than exhaustive search, but huge memory

Third-party cryptanalysis of KECCAK

Practical-complexity attacks on KECCAK

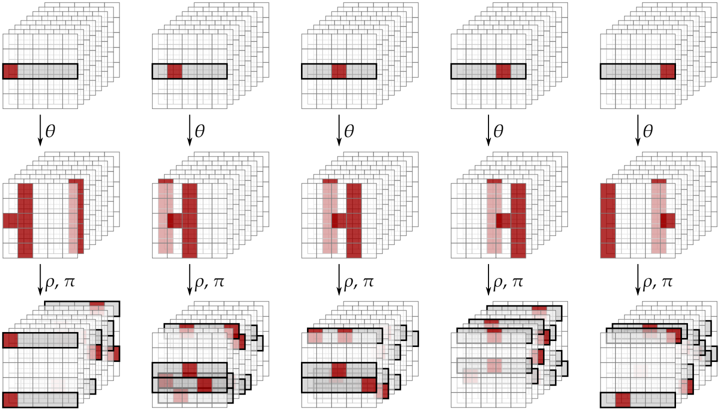
Rounds	
2	preimages and collisions [Morawiecki, CC]
2	collisions [Duc, Guo, Peyrin, Wei, FSE 2012 and CC]
3	40-bit preimage [Morawiecki, Srebrny, 2010]
3	near collisions [Naya-Plasencia, Röck, Meier, Indocrypt 2011]
4	key recovery [Lathrop, 2009]
4	distinguishers [Naya-Plasencia, Röck, Meier, Indocrypt 2011]
4	collisions [Dinur, Dunkelman, Shamir, FSE 2012 and CC]
5	near-collisions [Dinur, Dunkelman, Shamir, FSE 2012]

CC = Crunchy Crypto Collision and Preimage Contest

Observations from third-party cryptanalysis (1/2)

- Effect of **alignment** on differential/linear propagation
  - **Strong**: low uncertainty in prop. along block boundaries
  - Weak: high uncertainty in prop. along block boundaries
- Strong alignment puts barriers in the round function
- Weak alignment in KECCAK-*f*
  - strives to remove all such barriers
  - limits feasibility of rebound

Weak alignment, illustrated

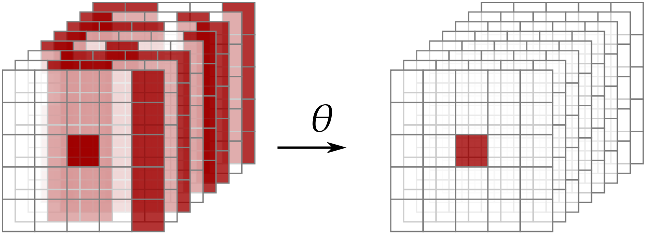


Basis for the possible output patterns of a single active row

Observations from third-party cryptanalysis (2/2)

- Effect of the **inverse** of the mixing layer θ
  - θ^{-1} is very dense
  - Limits the construction of high-probability trails over more than a few rounds

Inverse of θ, illustrated



Single active bit at θ output
$\downarrow$
About half of the bits active at θ input

Differential and linear cryptanalysis

Lower bound for the weight of differential or linear trails?

- ARX: no relevant bounds
- AES-based: strong and simply provable bounds, **but**
  - Not for truncated differentials and rebound attack
- Weak alignment: computer-assisted proofs are possible
  - Tight bound for 3 rounds of KECCAK-$f[1600]$
  - Lower bound for 6 rounds of KECCAK-$f[1600]$

Rounds	Best known diff. weight	
1	2	
2	8	
3	32	[Duc et al.]
4	134	[KECCAK team]
5	510	[Naya-Plasencia et al.]
6	1360	[KECCAK team]

Outline

1 Keccak uses a wide permutation

2 Keccak's safety margins

3 Keccak's cryptanalysis strengths

4 Keccak's offering

5 Conclusions

Modes of use

- Keccak is a **sponge** function
  - Hashing, stream encryption, MAC computation, full domain hashing, randomized hashing …
  - Variable-output length makes it suitable for tree hashing [ePrint 2009/210]
- Keccak is a **duplex** object
  - Reseedable pseudo-random bit generator
  - Authenticated encryption
- Unprecedented simplicity & flexibility
  - Exchange *rate* for *capacity*, and vice versa
  - Joint security of multiple instances [SAC 2011]

End-to-end approach

Remember, security is like a chain...

- Security of the mode
- Security of the primitive
- Security of the implementation (in a keyed mode)
  - Resistance against cache-timing attacks
  - Resistance against power/electromagnetic analysis
  - At a reasonable cost!

Diversity, diversity, **diversity**

- Choice of basic building blocks
  - MD5, SHA-1 and SHA-2: ARX
  - AES: byte-oriented, MDS mixing layer, 8-bit S-box
  - Keccak is bit-oriented and weakly aligned
- Choice of basic primitive
  - MD5, SHA-1 and SHA-2: block cipher based
  - AES: block cipher
  - Keccak uses an iterated permutation
- Choice of mode of use
  - MD5, SHA-1 and SHA-2: Merkle-Damgård, Davies-Meyer, MD-strengthening, HMAC, MGF1, …
  - AES: CBC, counter, C-MAC, GCM, CCM, …
  - **Keccak uses the sponge and duplex constructions**

Outline

1 KECCAK uses a wide permutation

2 KECCAK's safety margins

3 KECCAK's cryptanalysis strengths

4 KECCAK's offering

5 Conclusions

KECCAK has strong (and sometimes unique) features

- Design and security
  - Thick safety margin
  - Third-party cryptanalysis and bounds on differential trails
  - Matryoshka principle: cryptanalysis from small to large
  - Provable security against generic attacks
  - Diversity w.r.t. AES and SHA-1/-2 (ARX)
- Flexibility inherent in the sponge and duplex constructions
  - Simple security claim, disentangled from output length
  - Arbitrary output length (for, e.g., MGF, stream cipher)
  - Single permutation for all output lengths
  - Performance-security (rate-capacity) trade-offs
  - No output transformation (e.g., efficient duplexing)
- Implementation
  - Good software performance
  - Excellent suitability on hardware with speed/area trade-offs
  - Secure implementations much cheaper than other designs

Our references

- *Differential propagation in* Keccak*, FSE 2012*
- Keccak *implementation overview* (version 3.1 or later)
- KeccakTools (version 3.2 or later)
- *On alignment in* Keccak, Ecrypt II Hash Workshop 2011
- *The* Keccak *reference* (version 3.0 or later)
- *The* Keccak *SHA-3 submission*, 2011
- *Building power analysis resistant implementations of* Keccak, SHA-3 2010
- *Note on zero-sum distinguishers of* Keccak-*f*, NIST hash forum 2010
- *Note on* Keccak *parameters and usage*, NIST hash forum 2010
- *Note on side-channel attacks and their counterm...*, NIST hash forum 2009
- *The road from* Panama *to* Keccak *via* RadioGatún, Dagstuhl 2009

http://keccak.noekeon.org/

Our references

- *Duplexing the sponge: authenticated enc. and other applications, SAC 2011*
- *On the security of the keyed sponge construction, SKEW 2011*
- *Cryptographic sponge functions* (version 0.1 or later)
- *Sponge-based pseudo-random number generators*, CHES 2010
- *Sufficient conditions for sound tree and seq. hashing modes*, ePrint 2009
- *On the indifferentiability of the sponge construction*, Eurocrypt 2008
- *Sponge functions*, comment to NIST and Ecrypt Hash Workshop 2007

http://sponge.noekeon.org/