Radbound University

Sponge functions

Cryptography, Spring 2021

L. Batina, J. Daemen

March 9, 2021

Institute for Computing and Information Sciences
Radboud University

Outline

The SHA-3 competition

Sponge functions

KECCAK and SHA-3

The SHA-3 competition

Towards the SHA-3 competition

- 2005-2006: MD5 and SHA-1 crisis
  - actual collisions for MD5
  - theoretical collision attacks for SHA-1
  - attacks on Merkle-Damgård with higher success probability than believed up to that point
- SHA-2 based on same principles, so NIST got nervous
- 2007: NIST announces plans to have open SHA-3 competition
  - goal: find a worthy successor for SHA-2
  - similar process as AES competition
- 2008: NIST publishes SHA-3 requirements
  - *more efficient than SHA-2*
  - output lengths: 224, 256, 384, 512 bits
  - security: collision and (2nd) pre-image resistance strengths
  - specs, code, design rationale and preliminary analysis
  - patent waiver

The SHA-3 competition

- Started in 2008
- Three-round public process
  - round 1: 64 submissions, 51 accepted
  - round 2: 14 semi-finalists
  - round 3: 5 finalists
- All selections done by NIST but based on public evaluation by crypto community
- October 2012: NIST announces the SHA-3 winner
- The winner: KECCAK by Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche
  - something completely different than MD5/SHA-1/SHA-2
  - ...and completely different than Rijndael/AES
- August 2015: NIST finally publishes the SHA-3 standard: FIPS 202

Sponge functions

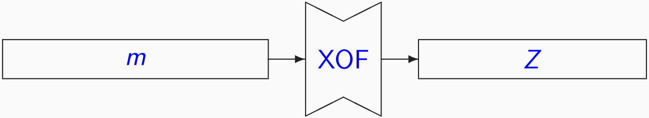
The idea: permutation-based hashing

KECCAK uses a construction called *sponge*, where did that come from?

- Our goal: find a hashing mode that is sound and simple
  - with birthday-bound $\mathcal{RO}$-differentiating advantage
  - calling a primitive that we know how to design
  - nice to have: support for arbitrary output length
- Block cipher as a primitive
  - round function design: several good approaches known
  - key schedule: not so clear how to do design good one
- But do we really need a block cipher?
  - no need for separation between data path and key schedule
  - let us merge them: *an (iterative) permutation*
- Result: *the sponge construction*

  [Bertoni, Daemen, Peeters, Van Assche (KECCAK team) 2007]

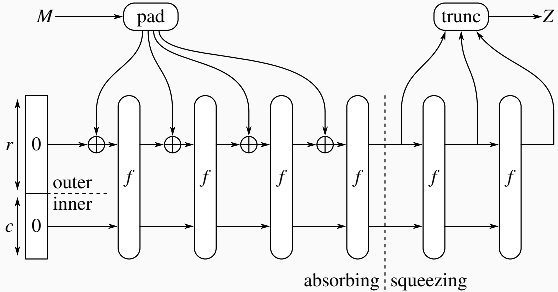
Modern hashing: Extendable Output Function (XOF)



$$Z = \mathrm{XOF}(m, n)$$

- Many use cases of hashing require outputs longer or shorter than some nominal digest length
- XOF:
  - user specifies output length n when calling the function
  - name introduced in SHA-3 standard [FIPS 202]
- *Secure if it behaves as a* $\mathcal{RO}$
- Strength specified in terms of (internal) parameter *capacity* c

The sponge construction

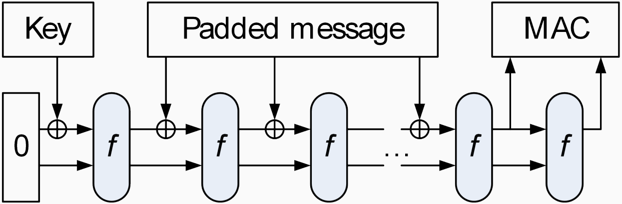


- Builds a XOF from a b-bit **permutation** f, with $b = r + c$
  - r bits of *rate*
  - c bits of *capacity* (security parameter)
- $\mathcal{RO}$-differentiating advantage $= N^2 2^{-(c+1)}$ [Keccak team, 2008]
  - due to collisions in c-bit inner part
  - super-tight: it is the birthday bound in c

Implications of the $\mathcal{RO}$-differentiating bound

- *Random sponge*: sponge construction with a random permutation $\mathcal{P}$
- Success probability of attack on random sponge upper bound by
  - success probability of that attack on $\mathcal{RO}$, plus
  - differentiating advantage of random sponge from $\mathcal{RO}$
- Classical attacks on random sponge with output truncated to n bits:
  - collision: $N^2 2^{-(n+1)} + N^2 2^{-(c+1)}$
  - (first) pre-image: $N 2^{-n} + N^2 2^{-(c+1)}$
  - 2nd pre-image resistance: $N 2^{-n} + N^2 2^{-(c+1)}$
- Security strength of random sponge truncated to n bits
  - collision-resistance: $\min(c/2, n/2)$
  - 1st or 2nd pre-image resistance: $\min(c/2, n)$
- But in reality we have to construct a concrete permutation f
  - above bounds are for *generic attacks*: those that do not exploit specific properties of f
- Once we fix f, we are again in the *world of cryptanalysis*

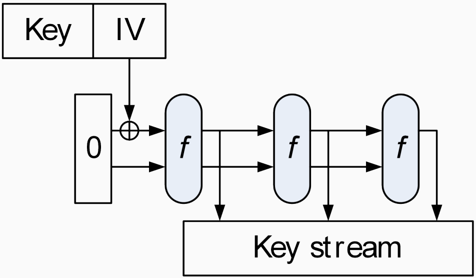
Using sponge for MAC (and key derivation)



- $\mathrm{MAC}_K(m) = \mathrm{XOF}(K \| m, n)$
- $\mathrm{KDF}_K(D) = \mathrm{XOF}(K \| D, n)$

No need for patches à la HMAC as sponge is basically sound

Stream cipher mode



- Many output blocks per D: similar to OFB
- 1 output block per D: similar to counter mode

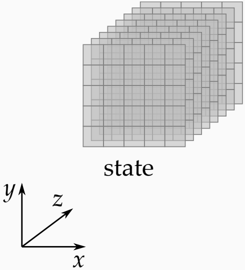
Note: figure indicates diversifier by IV

Required primitive: a cryptographic permutation

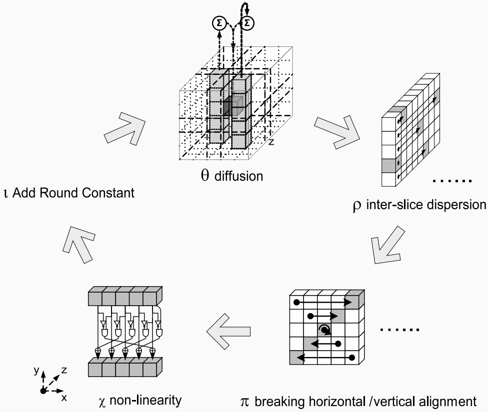
- A permutation that *should not have exploitable properties*
- Like a block cipher
  - sequence of identical rounds
  - round function is sequence of simple step mappings
- . . . but not quite
  - no key schedule
  - round constants instead of round keys
  - inverse permutation need not be efficient

Keccak and SHA-3

- KECCAK is a sponge function using permutation KECCAK-f
- KECCAK-f operates on 3-dimensional state:
  - 5×5 *lanes*, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
  - (5×5)-bit *slices*, 2^ℓ of them



Keccak-f: the steps of the round function



bit-oriented highly-symmetric *wide-trail* design

Keccak$[r, c]$

- Sponge function using the permutation KECCAK-f
  - 7 permutations: $b \in \{25, 50, 100, 200, 400, 800, 1600\}$

    ... from toy over lightweight to high-speed ...
- SHA-3 instance SHAKE128: $r = 1344$ and $c = 256$
  - permutation width: 1600
  - security strength 128
- Lightweight instance: $r = 40$ and $c = 160$
  - permutation width: 200
  - security strength 80: what SHA-1 should have offered
- Security status:
  - best attack on hash function covers 6-round version
  - $\#$ rounds ranges from 18 for $b = 200$ to 24 for $b = 1600$

See [The KECCAK reference] at `keccak.team` for details

The XOFs and hash functions in FIPS 202

- Four drop-in replacements for SHA-2 and two XOFs
- All use Keccak-f with $b = 1600$
- Domain-separated from each other:
  - padding rule ensures separation between different capacities c
  - XOF inputs end in **11**, drop-in inputs end in **01**
  - XOF Tree-hashing ready: Sakura encoding [ePrint 2013/231]

XOF	**SHA-2 drop-in replacements**
Keccak$[c = 256](m \| 11 \| \mathbf{11})$	
	first 224 bits of Keccak$[c = 448](m \| \mathbf{01})$
Keccak$[c = 512](m \| 11 \| \mathbf{11})$	
	first 256 bits of Keccak$[c = 512](m \| \mathbf{01})$
	first 384 bits of Keccak$[c = 768](m \| \mathbf{01})$
	first 512 bits of Keccak$[c = 1024](m \| \mathbf{01})$
SHAKE128 and **SHAKE256**	**SHA3-224** to **SHA3-512**

Conclusions

Conclusions

- Modern hashing is based on permutations
- SHA-3 is based on KECCAK
- Sponge construction: up to $c/2$ bits of security strength
- XOFs SHAKE128 and SHAKE256 can simplify usage
- No more need for HMAC or MGF1!
- All symmetric crypto can be based on permutations: **symmetric crypto 2.0**