Hashing and sponge functions
Part 2: What we can show and what we build

Joan Daemen[1]

Joint work with
Guido Bertoni[1], Michaël Peeters[2] and Gilles Van Assche[1]

[1]STMicroelectronics [2]NXP Semiconductors

NISNet Winter School,
Finse
May 26, 2011

Outline

1 Distinguishing a random sponge from a random oracle

2 Using the sponge construction for building functions

3 Soundness of the sponge construction

4 Applications

5 The duplex construction

6 Security proof for keyed modes

7 Conclusions

Outline

1 Distinguishing a random sponge from a random oracle

2 Using the sponge construction for building functions

3 Soundness of the sponge construction

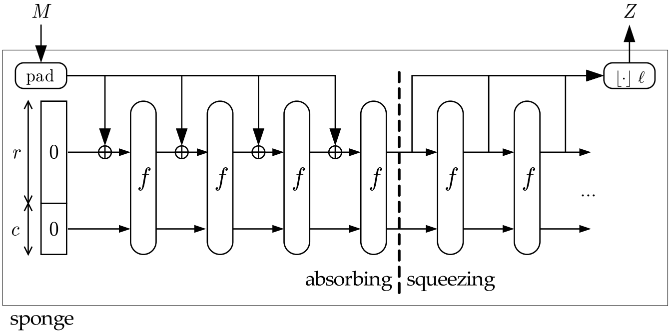
4 Applications

5 The duplex construction

6 Security proof for keyed modes

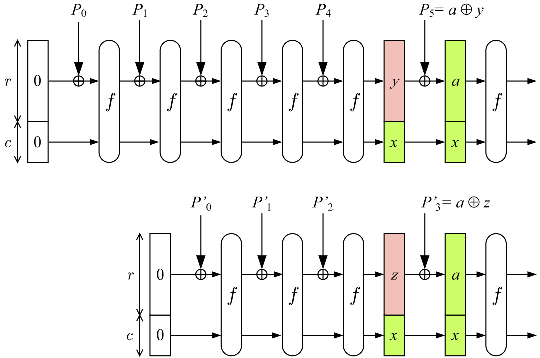
7 Conclusions

The sponge construction



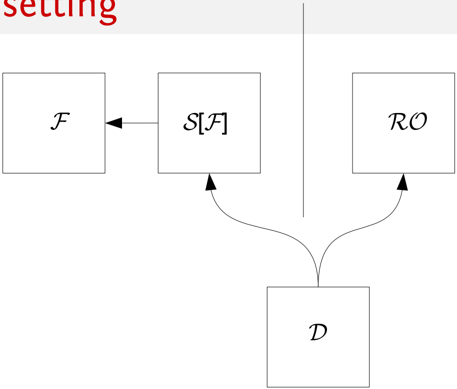
- b-bit state
  - *outer* part: top r bits
  - *inner* part: bottom c bits

State and inner collisions



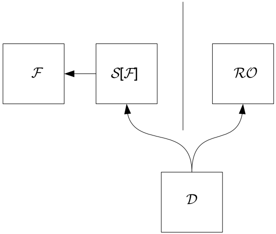
- State collision: different inputs leading to same state
- Inner collision: different inputs leading to same *inner* state

Distinguisher setting



- Adversary $\mathcal{D}$ is presented a system $\mathcal{X}$ that is either:
  - A random oracle $\mathcal{RO}$
  - A random sponge $\mathcal{S}[\mathcal{F}]$
- ...and must guess which one of the two $\mathcal{X}$ is

Distinguisher setting



- Adversary sends queries (M, ℓ) according to algorithm $\mathcal{A}$
- Success probability of correct guess: $\Pr(\text{success}|\mathcal{A})$
- Concept of advantage:

$$\Pr(\text{success}|\mathcal{A}) = \frac{1}{2} + \frac{1}{2}\text{Adv}(\mathcal{A})$$

- Express advantage as a function of total cost of queries N

A bound on the $\mathcal{RO}$-distinguishing advantage

- We define the cost of a query as: $N(M, \ell) = \lfloor \frac{|M|+1}{r} \rfloor + \lceil \frac{\ell}{r} \rceil$
- Equals $\#$ calls to $\mathcal{F}$ in case of random sponge
- Attack cost $N = \sum_i N(M_i, \ell_i)$ of all queries

$\mathcal{RO}$-distinguishing advantage bounding theorem

$$\mathrm{Adv}(\mathcal{A}) \leq \mathrm{Adv}_{\mathrm{max}} \leq \frac{N^2}{2^{c+1}}$$

- $=$ success probability of optimum inner-collision search
- As tight as theoretically possible

Implications of the distinguishing advantage bound

- Let $\mathcal{A}$: n-bit output pre-image attack. Success probability:
  - $\mathcal{X} = \mathcal{RO}$: $P_{\text{pre}}(\mathcal{A}|\mathcal{RO})$
  - $\mathcal{X} = \mathcal{S}[\mathcal{F}]$: $P_{\text{pre}}(\mathcal{A}|\mathcal{S}[\mathcal{F}])$
- It is easy to see that:
  - $P_{\text{pre}}(\mathcal{A}|\mathcal{S}[\mathcal{F}]) \leq \text{Adv}_{\text{max}} + P_{\text{pre}}(\mathcal{A}|\mathcal{RO})$
  - if not true, $\mathcal{A}$ would form a distinguisher with advantage:
  - $\text{Adv}(\mathcal{A}) = P_{\text{pre}}(\mathcal{A}|\mathcal{S}[\mathcal{F}]) - P_{\text{pre}}(\mathcal{A}|\mathcal{RO}) > \text{Adv}_{\text{max}}$
- This can be generalized to any attack
  - Upper bounds success probability of all generic attacks
  - Justifies flat sponge claim!

Outline

1 Distinguishing a random sponge from a random oracle

2 Using the sponge construction for building functions

3 Soundness of the sponge construction

4 Applications

5 The duplex construction

6 Security proof for keyed modes

7 Conclusions

Using the sponge construction in practice

- Up to now, we used random sponges as security reference
  - for expressing security claims and requirements
  - it appears that these claims can be met
- How to build functions for which such a claim can hold?
- Patch existing constructions
  - Merkle-Damgård is not sound but can be patched
  - for infinite output: mask generating function (MGF) mode
  - solutions are ugly and sub-optimal
- Use the sponge construction itself!
  - just design a suitable permutation f: known methods
  - distinguish sponge parameters r, c from claimed c

Design approach

- Hermetic sponge strategy
  - instantiate sponge function with some concrete f and c
  - have a flat sponge claim with the chosen c

Mission

Design permutation f without exploitable properties

Outline

1 Distinguishing a random sponge from a random oracle

2 Using the sponge construction for building functions

3 Soundness of the sponge construction

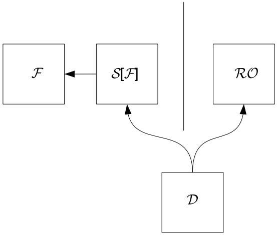
4 Applications

5 The duplex construction

6 Security proof for keyed modes

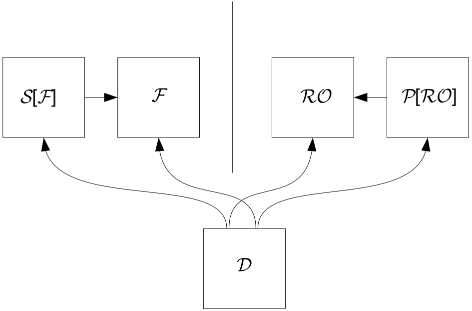
7 Conclusions

Is the sponge construction sound?



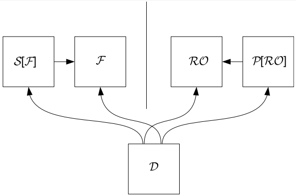
- Sponge construction is sound in this setting:
  - Proven bound: $\mathrm{Adv}_{\mathrm{max}} \leq \frac{N^2}{2^{c+1}}$
  - Imposes upper bound on success probability of any attack
- But the setting itself is not realistic!
  - Adversary $\mathcal{D}$ has no access to $\mathcal{F}$
  - In reality $\mathcal{F}$ is a publically specified f

Adapting the setting to reflect reality



- Adversary now has additional query access to $\mathcal{F}$ at the left
- But interfaces of left and right systems must match
  - Additional component at the right: $\mathcal{P}$
  - $\mathcal{P}$ is supposed to be hard to distinguish from $\mathcal{F}$

The indifferentiability framework



- Indifferentiability framework: Maurer et al.(2004)
  - Covers adversary with access to internal state at left
  - Additional interface, covered by a *simulator* at right
  - Applied to hash functions: Coron et al.(2005)
- Methodology:
  - Build $\mathcal{P}$ that makes left/right distinguishing difficult
  - Prove bound for advantage given this simulator $\mathcal{P}$
  - $\mathcal{P}$ may query $\mathcal{RO}$ for acting $\mathcal{S}$-consistently: $\mathcal{P}[\mathcal{RO}]$

The bound on the $\mathcal{RO}$-differentiating advantage

$\mathcal{RO}$-differentiating advantage bounding theorem

$$\mathsf{Adv}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{max}} \leq \frac{N^2}{2^{c+1}}$$

- Equal to $\mathcal{RO}$-distinguishing advantage bound
- Upper bounds success probability of any generic attack
- ...even for an adversary with access to f and f^{-1}
- Conclusion: the sponge construction is sound

Outline

1 Distinguishing a random sponge from a random oracle

2 Using the sponge construction for building functions

3 Soundness of the sponge construction

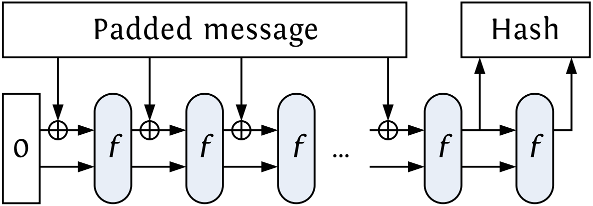
4 Applications

5 The duplex construction

6 Security proof for keyed modes

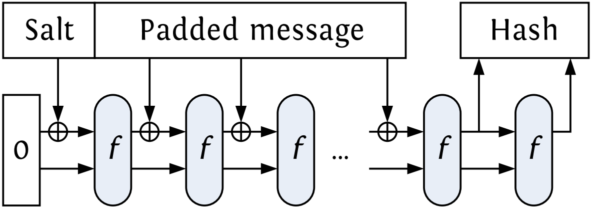
7 Conclusions

How to use a sponge function?



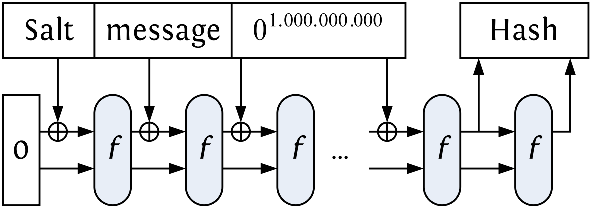
- For regular hashing

How to use a sponge function?



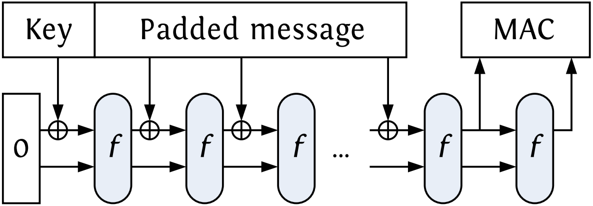
- For salted hashing

How to use a sponge function?



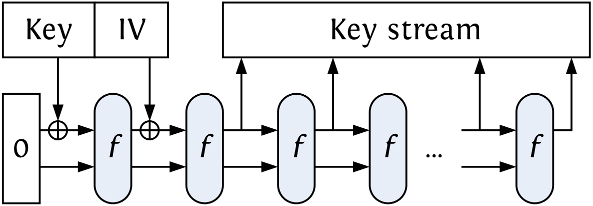
- For salted hashing, as slow as you like it

How to use a sponge function?



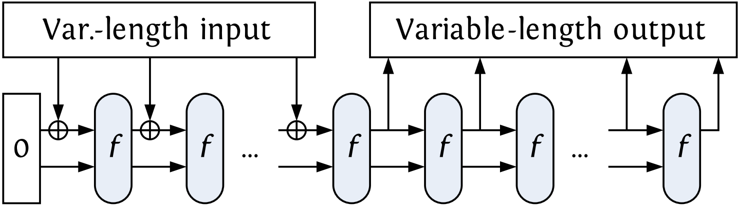
- As a message authentication code

How to use a sponge function?



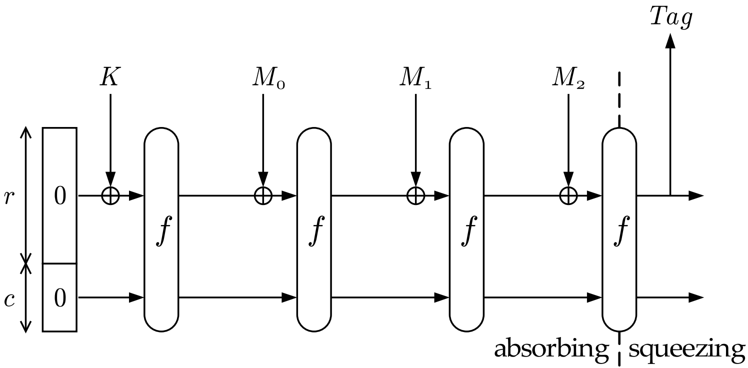
- As a stream cipher

How to use a sponge function?

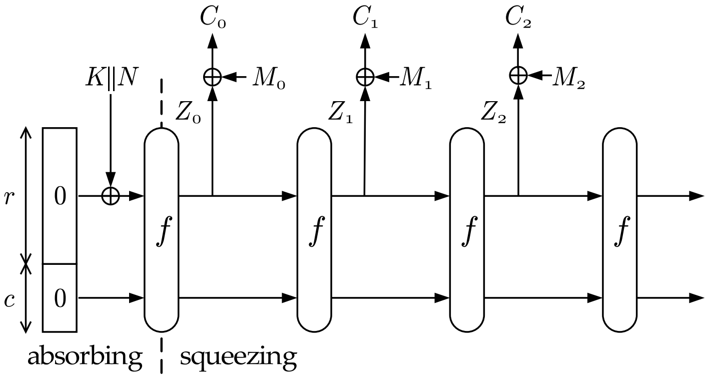


- As a mask generating function [PKCS#1, IEEE Std 1363a]

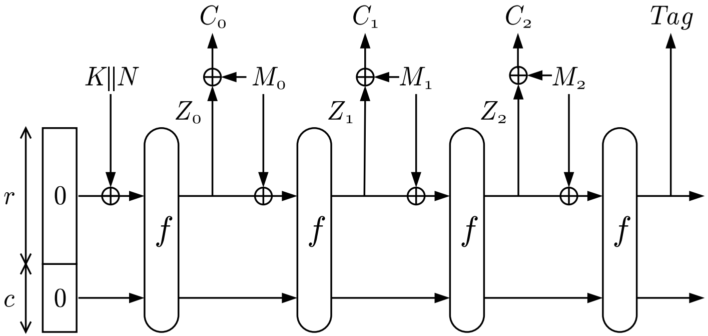
MAC generation with a sponge



Encryption with a sponge



Both encryption and MAC?



Absorbing and squeezing

Outline

1 Distinguishing a random sponge from a random oracle

2 Using the sponge construction for building functions

3 Soundness of the sponge construction

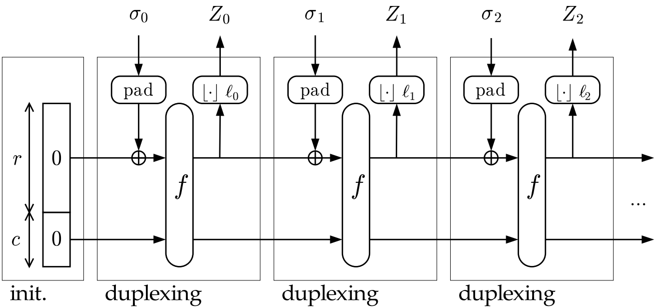
4 Applications

5 The duplex construction

6 Security proof for keyed modes

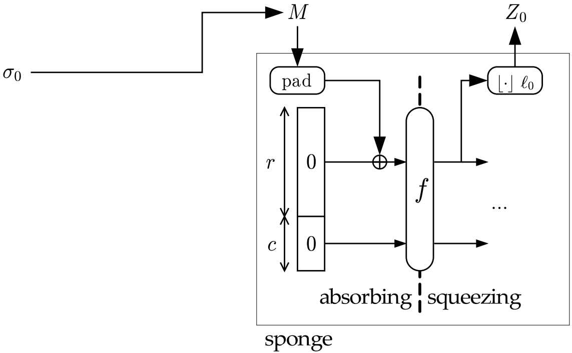
7 Conclusions

The duplex construction



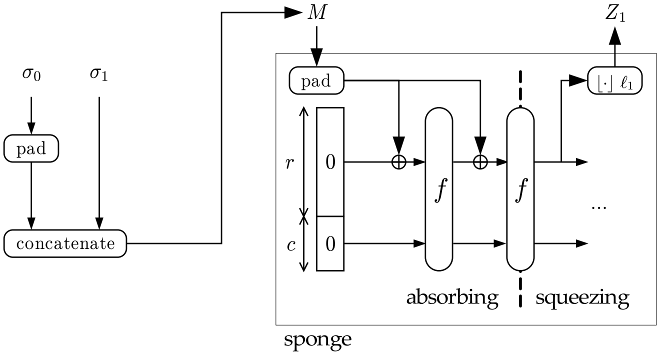
- Object: $D = \text{DUPLEX}[f, \text{pad}, r]$
- Requesting ℓ-bit output $Z = D.\text{duplexing}(\sigma, \ell)$
  - input σ and output Z limited in length
  - Z depends on all previous inputs

Generating duplex responses with a sponge



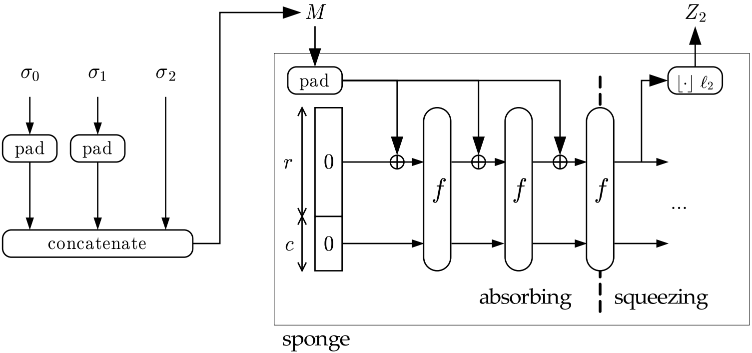
$$Z_0 = \mathsf{sponge}(\sigma_0, \ell_0)$$

Generating duplex responses with a sponge



$$Z_1 = \mathsf{sponge}(\mathsf{pad}(\sigma_0)||\sigma_1, \ell_1)$$

Generating duplex responses with a sponge



$$Z_2 = \mathsf{sponge}(\mathsf{pad}(\sigma_0)||\mathsf{pad}(\sigma_1)||\sigma_2, \ell_2)$$

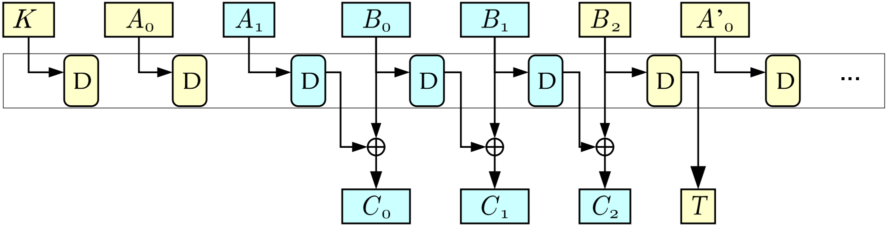
Properties of duplex construction

- Security of DUPLEX[f, pad, r] equivalent to SPONGE[f, pad, r]
- New type of *cryptographic object*
  - Input can be provided in each call
  - Output can be requested for each call
  - Memory: output to a call depends on all previous inputs
- Almost as efficient as the sponge construction itself
- Multi-rate security
  - Maximum length of σ two bits shorter than rate
  - For avoiding misalignment, add two bits to rate
  - Theorem: security of sponges sharing f with different c
- Opens up new applications …

Authenticated encryption

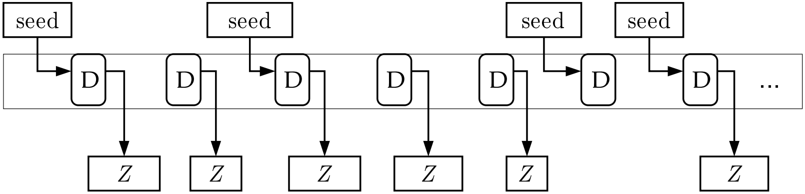
- Functionality:
  - Tag computation over data header and data body
  - Encryption of body into cryptogram, *diversified by* header
- Wrapping:
  - Input: key, data header and body
  - Output: tag and cryptogram
- Unwrapping
  - Input: key, data header and cryptogram, tag
  - Output: cryptogram or error message if tag is invalid
- Security requirements
  - Tag forgery infeasibility
  - Plaintext recovery infeasibility

The SpongeWrap mode



- Key K, data header A and data body B of arbitrary length
- Supports intermediate tags

Reseedable pseudorandom bit generator



Requirements:

- Seeding and reseeding
- Pseudo-random output depends on all past seeds
- Forward secrecy

Outline

1 Distinguishing a random sponge from a random oracle

2 Using the sponge construction for building functions

3 Soundness of the sponge construction

4 Applications

5 The duplex construction

6 Security proof for keyed modes

7 Conclusions

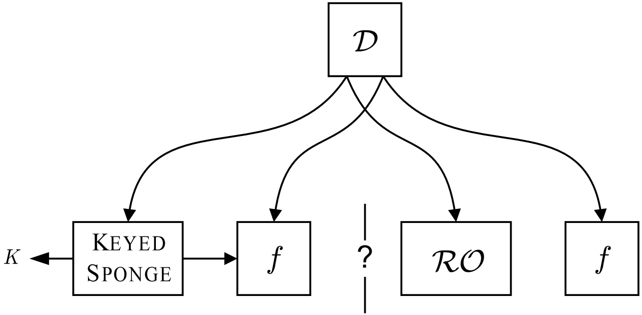
Keyed sponge functions

Keyed sponge

$$\textsc{KeyedSponge}[K](x) = \textsc{sponge}(K||x)$$

- E.g., $\text{MAC} = \textsc{KeyedSponge}(m)$

The adversary's setting



- M: online **data** complexity (blocks)
    - Calls to KEYEDSPONGE$[K]$ with unknown key K, or to $\mathcal{RO}$
- N: offline **time** complexity (calls to f)
    - Not involving the key

Distinguishing theorem

Upper bound on distinguishing advantage

$$\frac{M^2/2 + 2MN}{2^c} + P_{\text{key}}(N)$$

- $P_{\text{key}}(N)$: probability of guessing the key after N calls to f

If $M \ll 2^{c/2}$

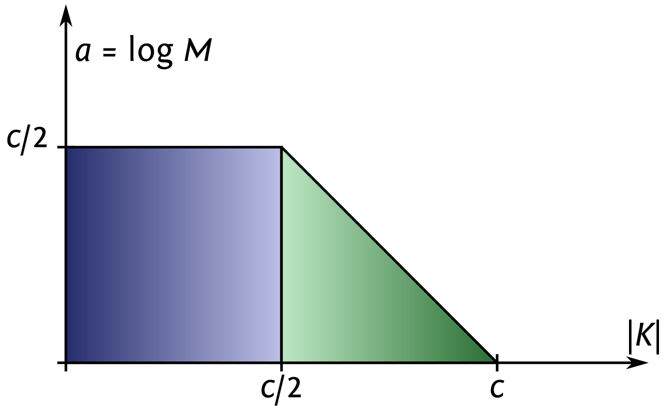
Time complexity is about $\min(2^{c-1}/M, 2^{|K|})$

Limited data complexity

- If the (online) data complexity is limited to $M \leq 2^a$
  - … by the protocol, by the secure device …
- And the capacity is $c \geq |K| + a + 1$
- Then we get the security of exhaustive key search

$$\min(2^{c-1}/M, 2^{|K|}) = 2^{|K|}$$

The new bound, illustrated



Building lightweight implementations

- Trade-off between security (c) and efficiency (r)
  - $b = r + c$
- Example 1: QUARK [Aumasson et al., QUARK, ..., CHES 2010]

U-QUARK	$r = 8$	$c = 128$
D-QUARK	$r = 16$	$c = 160$
S-QUARK	$r = 32$	$c = 224$

- Example 2: KECCAK supports : $b \in \{25, 50, 100 \ldots 1600\}$
  - E.g., KECCAK$[r = 40, c = 160]$ is compact in hardware
    [Bertoni et al., KECCAK implementation overview]

Building implementations that are even lighter

Target example: 80-bit key with QUARK

- Old bound: D-QUARK ($r = 16$, $c = 160$)
  - $c = 2|K|$
- New bound: U-QUARK ($r = 8$, $c = 128$)
  - with data complexity restricted to 2^{47} blocks

Outline

1 Distinguishing a random sponge from a random oracle

2 Using the sponge construction for building functions

3 Soundness of the sponge construction

4 Applications

5 The duplex construction

6 Security proof for keyed modes

7 Conclusions

Conclusions

- The flat sponge claim makes sense
- Sponge construction suitable for building secure primitive
- Sponge functions cover most symmetric crypto operations
- Duplex construction covers
  - efficient authenticated encryption
  - reseedable PRG
  - ...
- Bound for keyed modes allows lightweight sponges
- Sponge and duplex are just modes of a permutation
  - Do we still need hash functions, block- or stream ciphers?