On the Indifferentiability of the Sponge Construction

Guido Bertoni, Joan Daemen,
Michaël Peeters* and Gilles Van Assche
**NXP Semiconductors* *STMicroelectronics*

Introduction

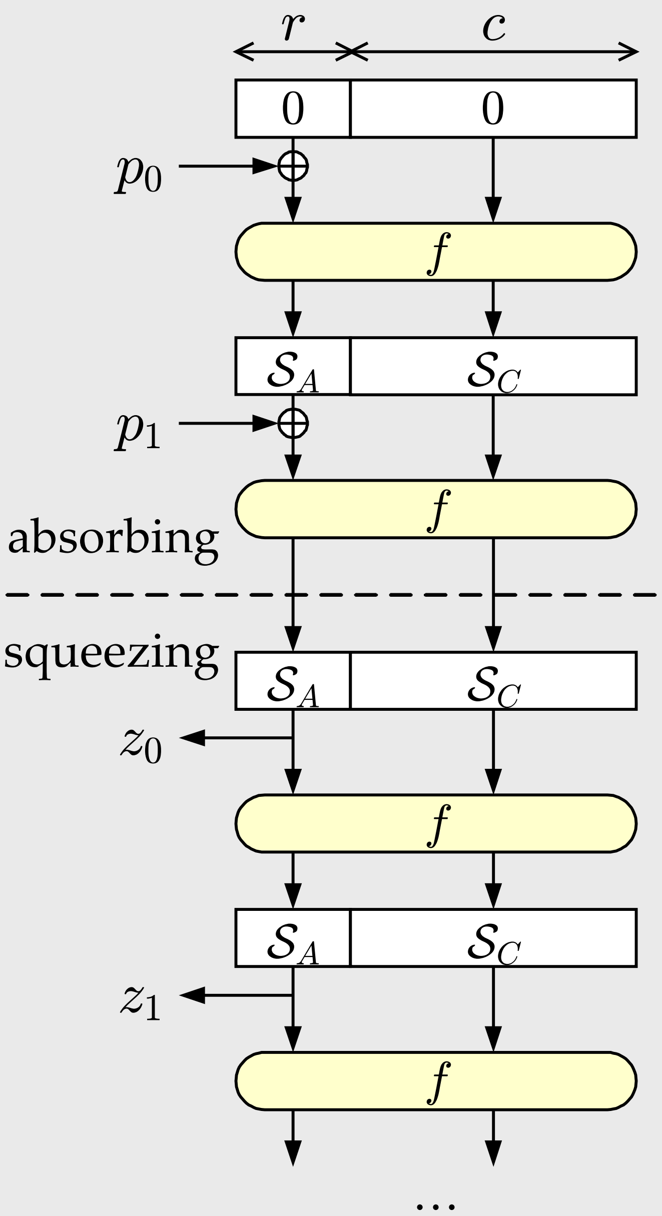
- Sponge functions model
  - the **finite state** of iterated cryptographic functions
    - as in iterated hash functions, stream ciphers, etc.
- Random sponges can be used
  - as a reference for (hash function) design
  - as an inspiration for (hash function) design
- Sponges are **simple**

Outline

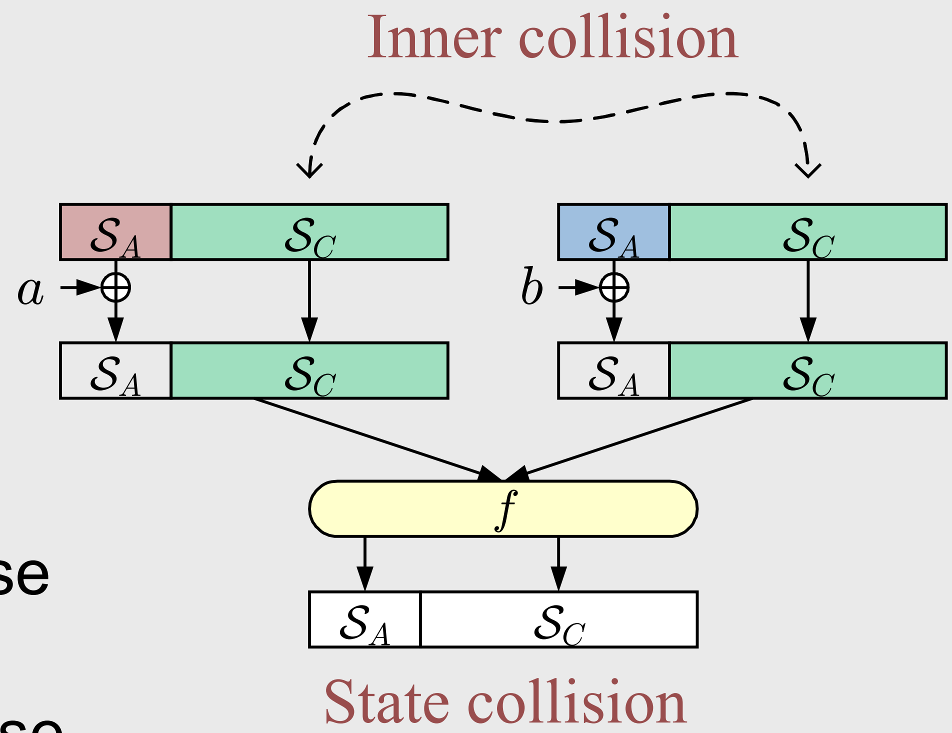
- ✓ Introduction
- ❑ **Definitions**
- ❑ Uses Examples
- ❑ Indifferentiability
- ❑ Constructing a sponge function
- ❑ Conclusion

Sponge Construction

The last absorbed block must not be zero.



Inner Collisions, State Collisions



- State collision
  - Absorbing phase
    - Hash collision
  - Squeezing phase
    - Output periodicity

Random Sponges

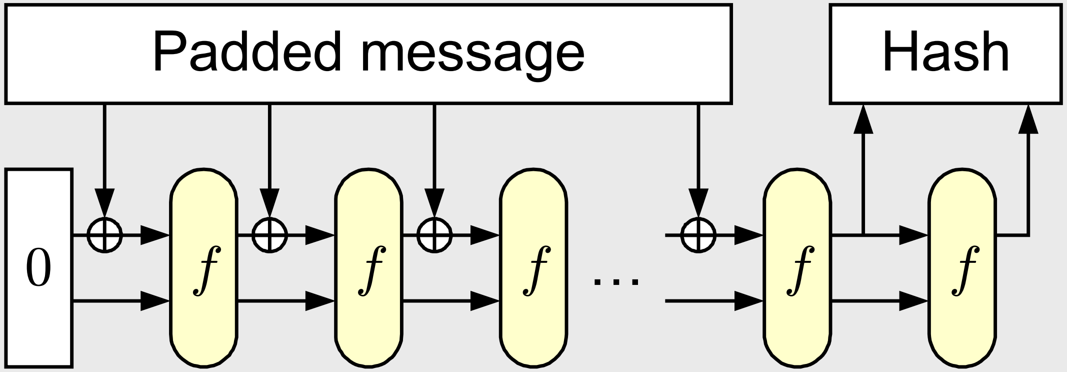
- Random T-sponge
  - Randomly chosen in $(2^{c+r})^{2^{c+r}}$ **transformations** f
- Random P-sponge
  - Randomly chosen in $(2^{c+r})!$ **permutations** f

Outline

- ✓ Introduction
- ✓ Definitions
- ❑ **Uses Examples**
- ❑ Indifferentiability
- ❑ Constructing a sponge function
- ❑ Conclusion

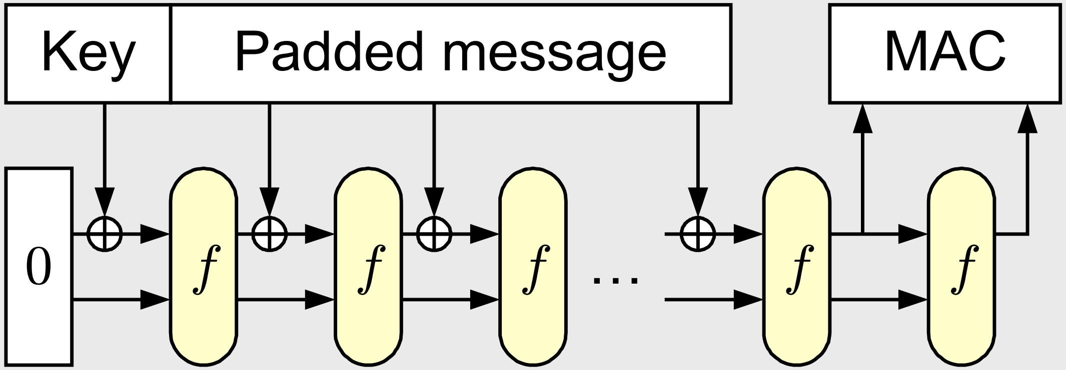
Uses Examples (1/5)

- Hash function



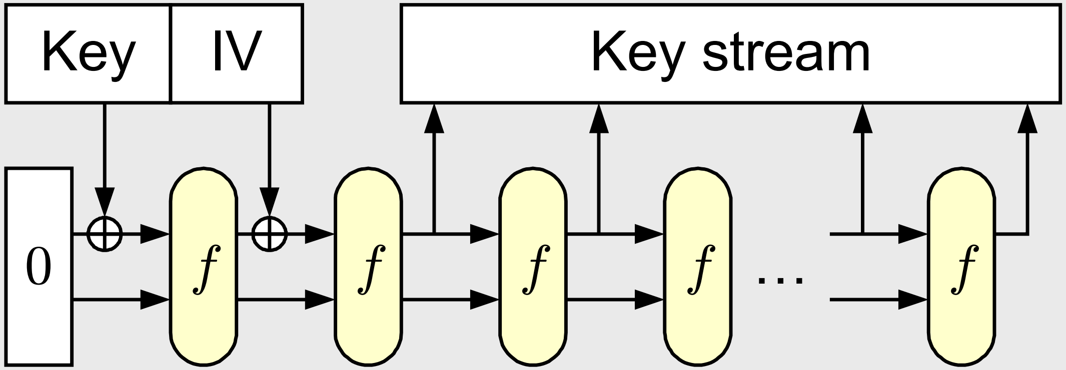
Uses Examples (2/5)

- Message authentication code



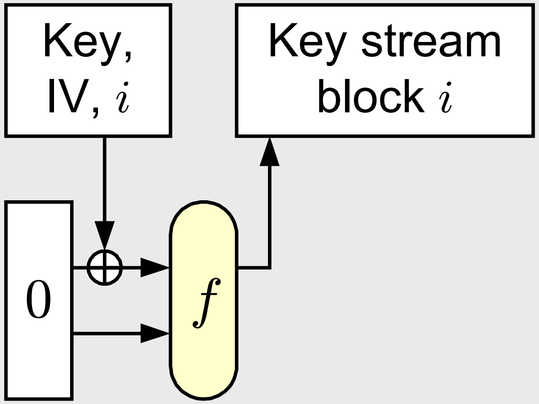
Uses Examples (3/5)

- Stream cipher



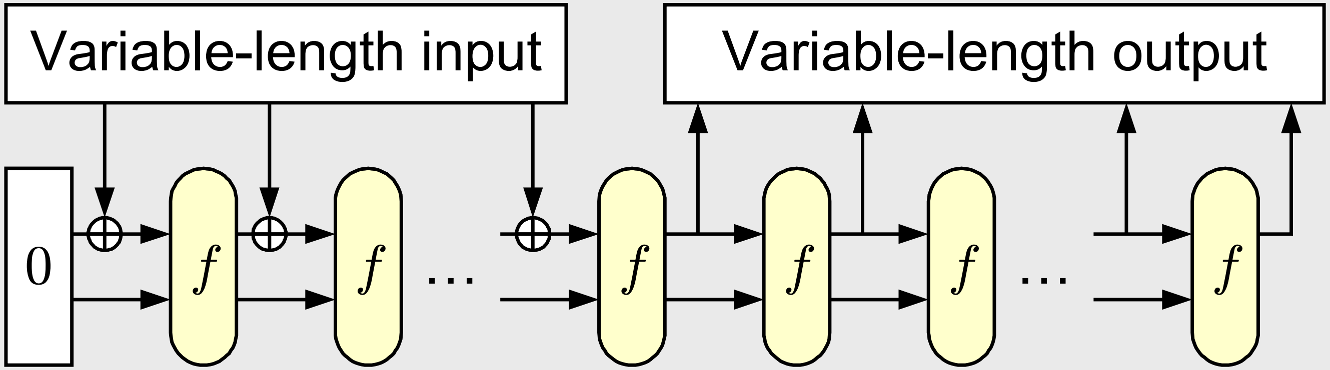
Uses Examples (4/5)

- Random-access stream cipher



Uses Examples (5/5)

- Mask generating functions, key derivation

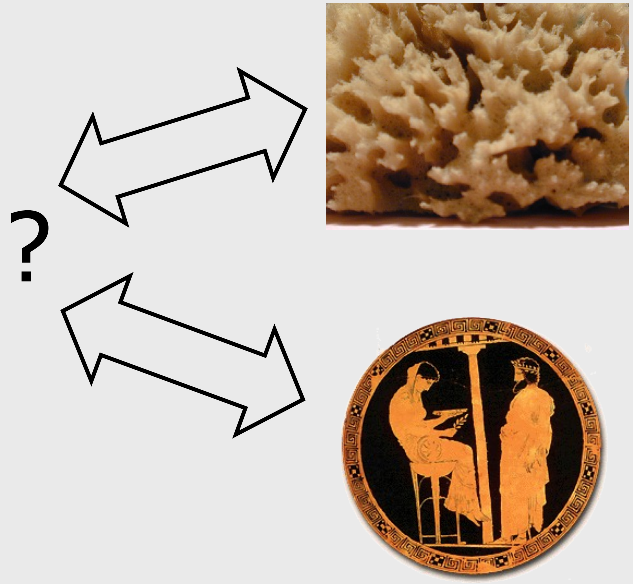


- See PKCS#1 and IEEE Std 1363a

Outline

- ✓ Introduction
- ✓ Definitions
- ✓ Uses Examples
- ❑ **Indifferentiability**
- ❑ Constructing a sponge function
- ❑ Conclusion

Distinguishing Random Sponges



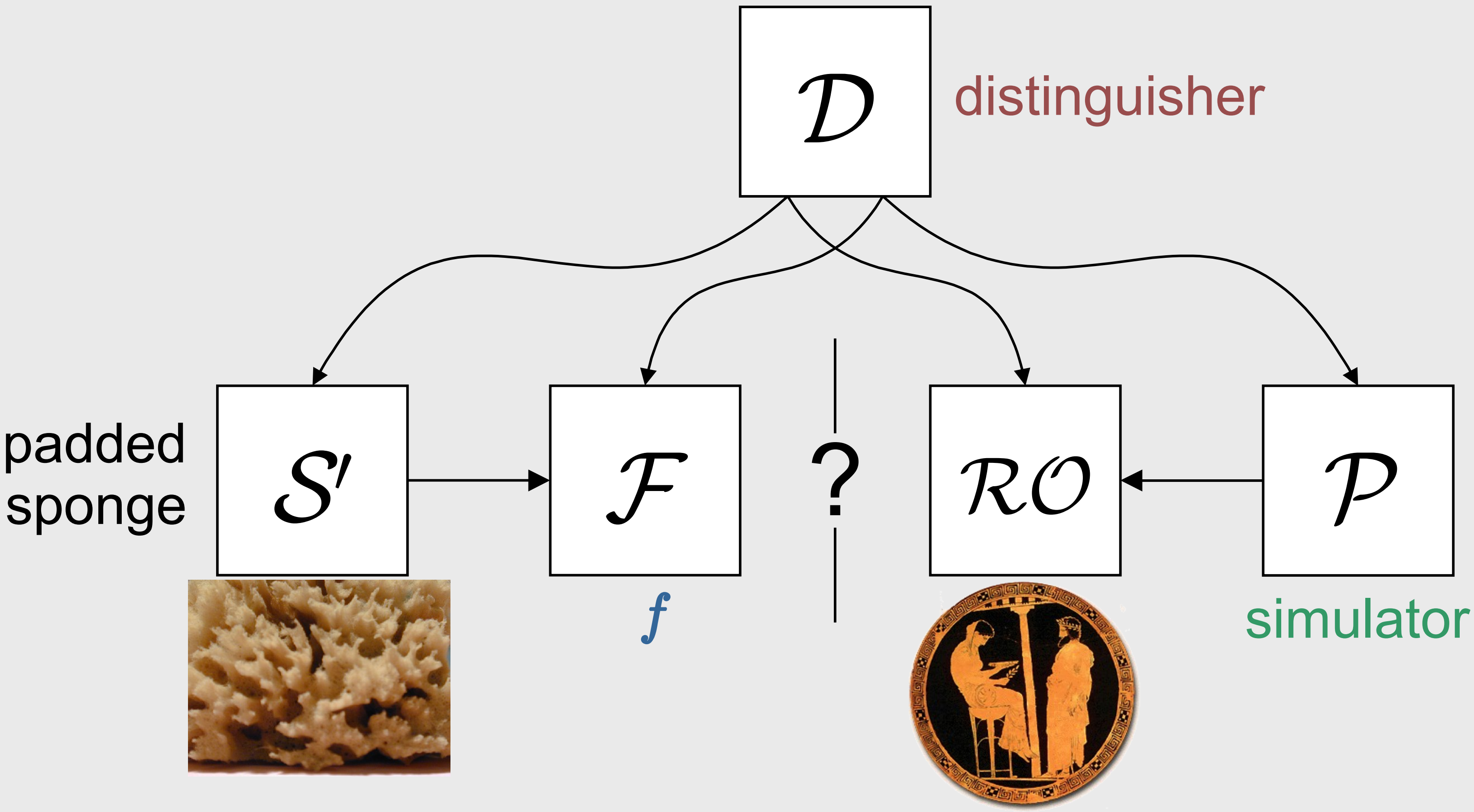
- Adversary queries a **black box**, either RS or RO
  - Budget of N input and output blocks
- **Theorem:** A random sponge can only be distinguished from a random oracle by the presence of **inner collisions**.
  - When $N \ll 2^{c/2}$, inner collisions are unlikely

Only gives an upper bound.

Indifferentiability Framework

- Goal: obtain **lower bound** on generic attacks
- Distinguisher has to differentiate between:
  - the ideal system (Random Oracle), and
  - the construction (here, the Sponge),
  - **with access** to publicly-known function or parameter (here, the transformation *f*)
- If indifferentiable
  - cryptosystem using construction as strong as cryptosystem using ideal system
- Maurer et al., TCC 2004; Coron et al., CRYPTO 2005

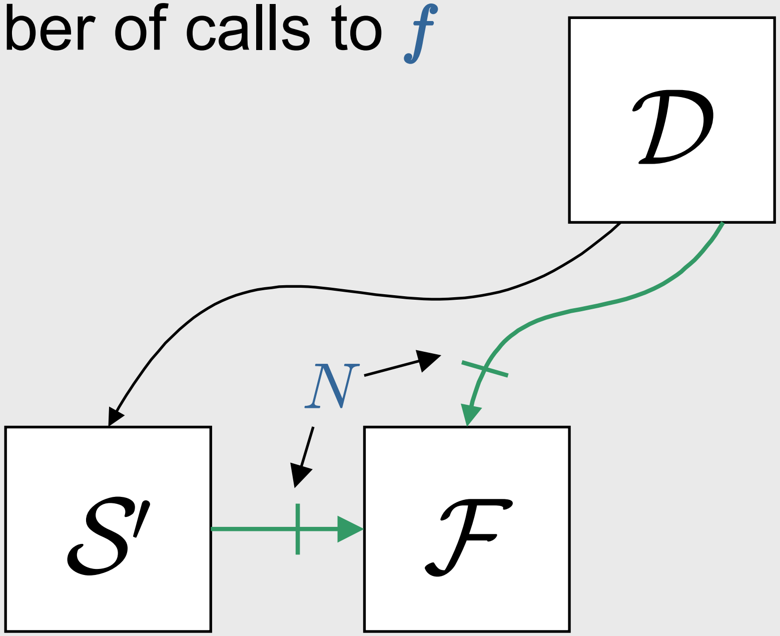
Differentiating Random Sponges



Differentiating Random Sponges

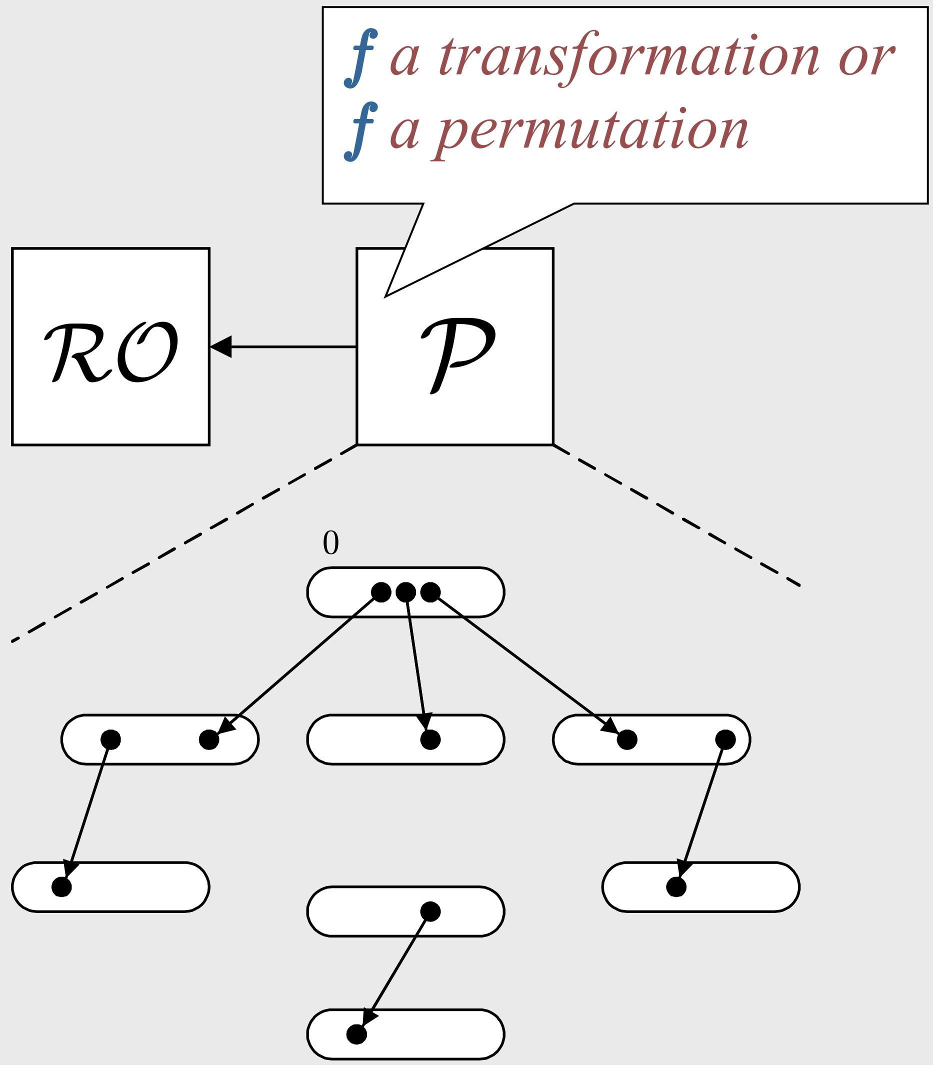
- **Theorem:** a random sponge can be differentiated from a random oracle only with probability $\approx N(N+1)/2^{c+1}$, with $N<2^c$.
  - Here N is the **total** number of calls to f

Generic attacks require $2^{c/2}$.



Proving the Indifferentiability

- Simulating f
  - Keeps memory of (input, output) pairs in a graph
- Properties
  - **Sponge-consistence** with what RO says
  - Similar **output distribution**
- Can be differentiated
  - By different distribution of simulator and random f

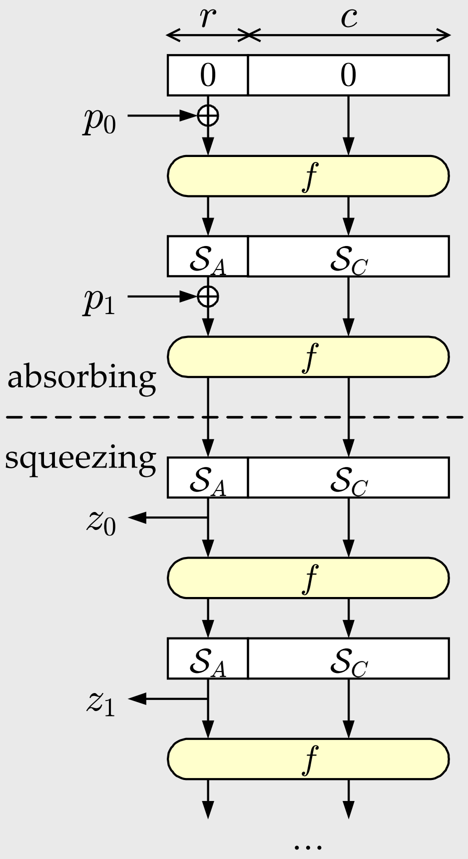


Outline

- ✓ Introduction
- ✓ Definitions
- ✓ Uses Examples
- ✓ Indifferentiability
- ❑ **Constructing a sponge function**
- ❑ Conclusion

Constructing a Sponge Function

- Choose c, r
  - No generic attacks below $2^{c/2}$
  - Transformation or permutation over $c+r$ bits
- Construct a *random*(!) transformation?
- Construct a *random*(!) permutation!
  - It shall not have any special properties(!)
    - except its compact description
  - Other constructions build upon permutations: see also Snefru, FFT-Hash, SMASH, …



Advantages of the Sponge Construction

- Relative simplicity in design
  - Permutation similar to block cipher design
    - E.g., block cipher without key schedule
- Flexibility
  - One permutation can accommodate for several (c,r) pairs
- Efficiency
- Simplicity
  - Simple model, simple proofs
  - Suitable for many applications
  - Variable-length output

Outline

- ✓ Introduction
- ✓ Definitions
- ✓ Uses Examples
- ✓ Indifferentiability
- ✓ Constructing a sponge function
- ❑ **Conclusion**

Conclusion

- Sponges are a simple model
  - to model the finite state of iterated primitives
- Sponges are a simple tool
  - for building hash functions and stream ciphers
  - for expressing compact security claims
- Sponges are fun!

http://sponge.noekeon.org/