Overview of the Sponge, Duplex and Farfalle constructions

Gilles Van Assche[1]

[1]STMicroelectronics

Summer school on real-world crypto and privacy
Šibenik, Croatia, June 2019

Based on joint work with Elena Andreeva, Guido Bertoni, Joan Daemen, Seth Hoffert, Bart Mennink, Michaël Peeters, Ronny Van Keer

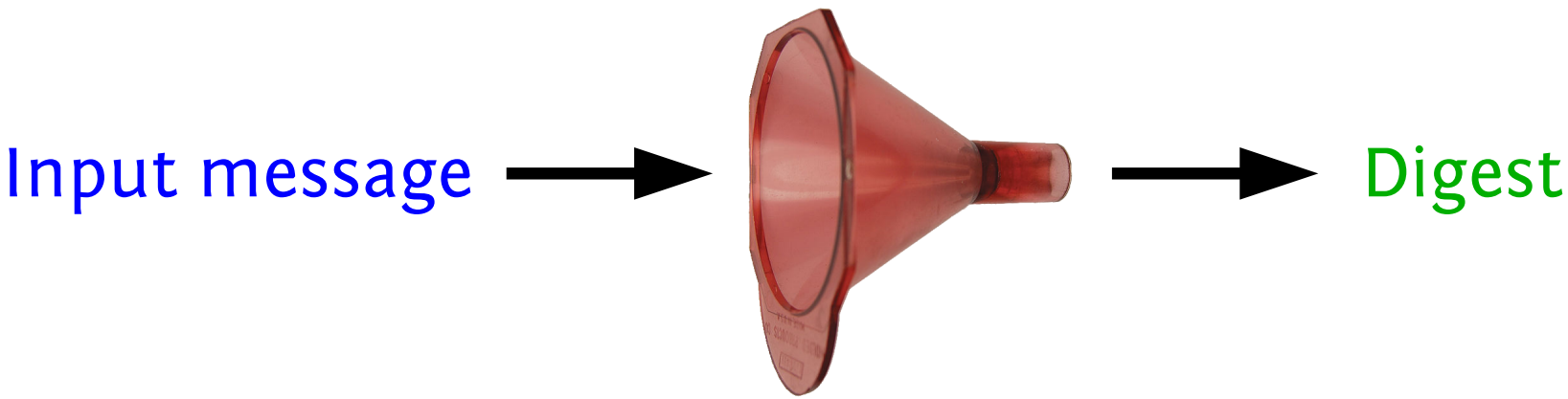
Outline

Outline

Outline

Cryptographic hash functions

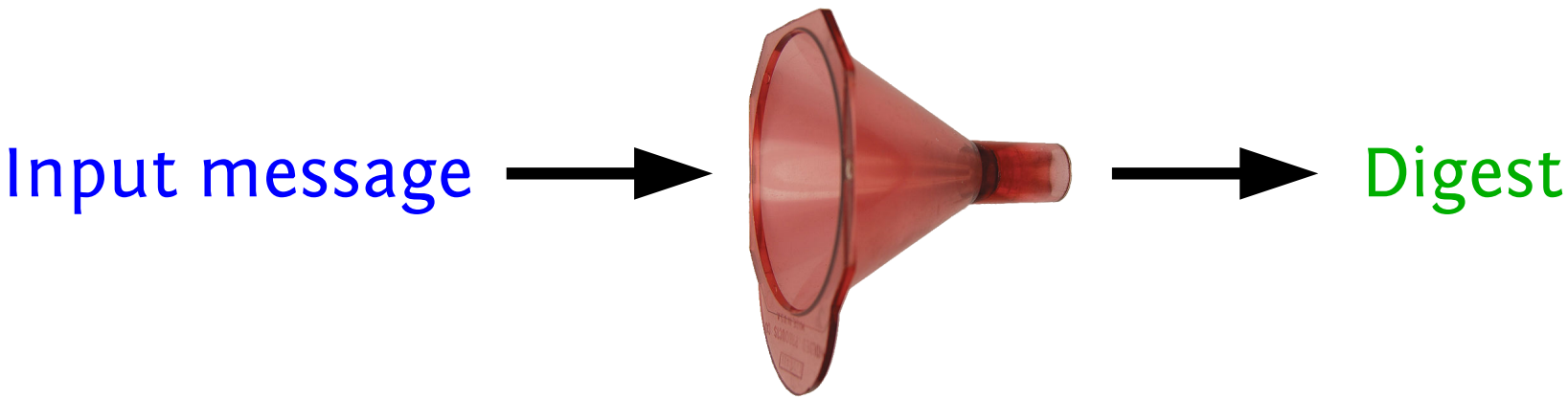
$$h : \{0,1\}^* \to \{0,1\}^n$$



- Applications
  - *Signatures*: $\text{sign}_{\text{RSA}}(h(M))$ instead of $\text{sign}_{\text{RSA}}(M)$
  - *Key derivation*: master key K to derived keys ($K_i = h(K\|i)$)
  - *Bit commitment, predictions*: h(what I know)
  - *Message authentication*: $h(K\|M)$
  - ...

Cryptographic hash functions

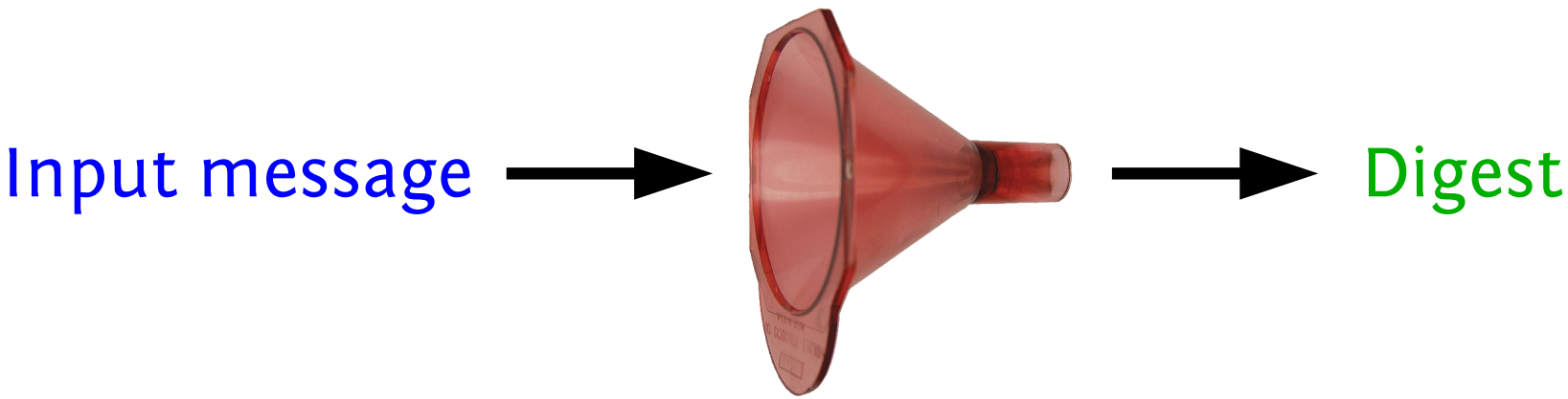
$$h : \{0,1\}^* \to \{0,1\}^n$$



- Applications
  - *Signatures*: $\text{sign}_{\text{RSA}}(h(M))$ instead of $\text{sign}_{\text{RSA}}(M)$
  - *Key derivation*: master key K to derived keys $(K_i = h(K\|i))$
  - *Bit commitment, predictions*: h(what I know)
  - *Message authentication*: $h(K\|M)$
  - ...

Cryptographic hash functions

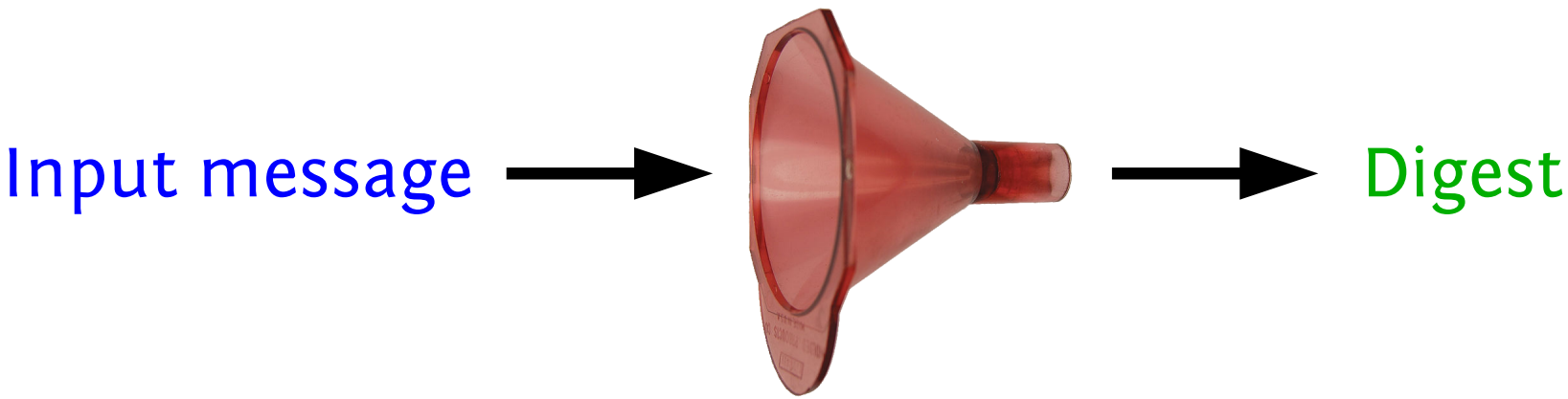
$$h : \{0,1\}^* \to \{0,1\}^n$$



- Applications
  - *Signatures*: $\text{sign}_{\text{RSA}}(h(M))$ instead of $\text{sign}_{\text{RSA}}(M)$
  - *Key derivation*: master key K to derived keys $(K_i = h(K\|i))$
  - *Bit commitment*, *predictions*: h(what I know)
  - *Message authentication*: $h(K\|M)$
  - ...

Cryptographic hash functions

$$h : \{0,1\}^* \to \{0,1\}^n$$



- Applications
  - *Signatures*: $\text{sign}_{\text{RSA}}(h(M))$ instead of $\text{sign}_{\text{RSA}}(M)$
  - *Key derivation*: master key K to derived keys $(K_i = h(K\|i))$
  - *Bit commitment*, *predictions*: $h(\text{what I know})$
  - *Message authentication*: $h(K\|M)$
  - ...

Generalized: extendable output function (XOF)

$$h : \{0,1\}^* \to \{0,1\}^*$$

"XOF: a function in which the output can be extended to any length."

[Ray Perlner, SHA-3 workshop 2014]

- Applications
  - *Signatures*: full-domain hashing, mask generating function
  - *Key derivation*: as many/long derived keys as needed
  - *Stream cipher*: $C = P \oplus h(K \| \text{nonce})$

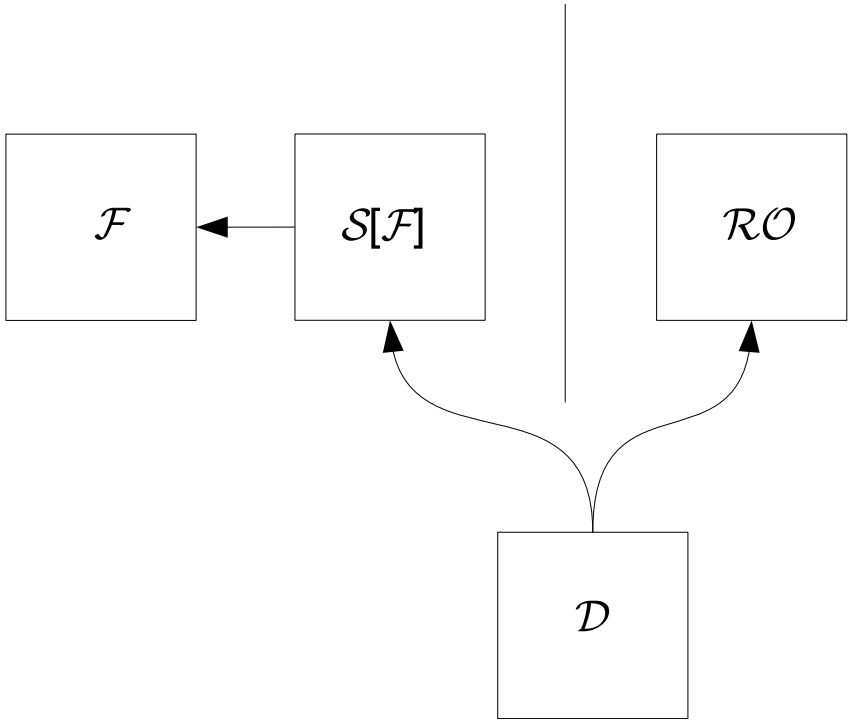
Modern security requirements

- Hash or XOF h with n-bit output
- Modern security requirements
  - h behaves like a random mapping
  - ... up to security strength s
- Classical security requirements, derived from it

Preimage resistance	$2^{\min(n,s)}$
Second-preimage resistance	$2^{\min(n,s)}$
Collision resistance	$2^{\min(n/2,s)}$

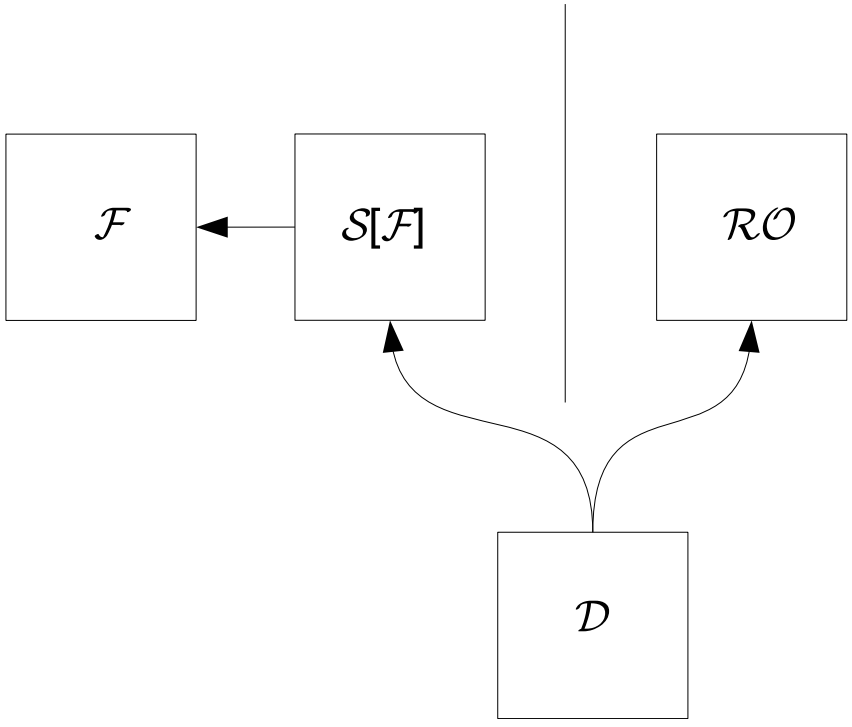
Outline

Generic security: indistinguishability



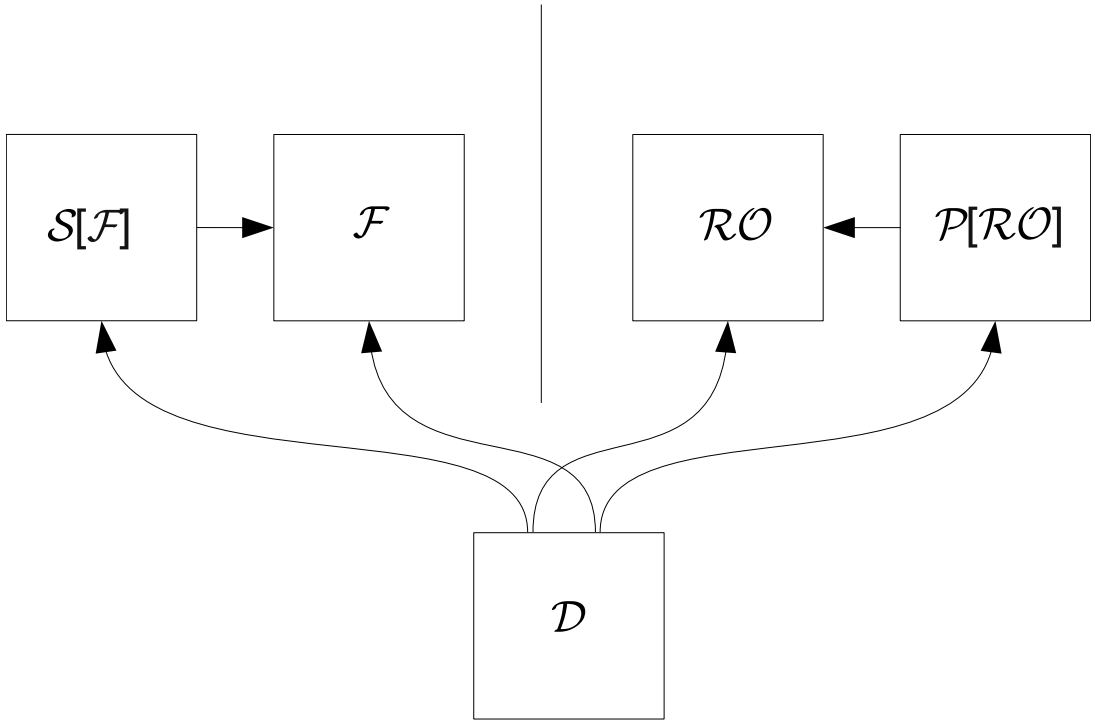
- Adversary $\mathcal{D}$ must tell apart
  - the ideal function: a monolithic random oracle $\mathcal{RO}$
  - construction $\mathcal{S}[\mathcal{F}]$ calling an ideal primitive $\mathcal{F}$
- Express $\Pr(\text{success}|\mathcal{D})$ as a function of total cost of queries N
- Problem: in real world, $\mathcal{F}$ is available to adversary

Generic security: indistinguishability



- Adversary $\mathcal{D}$ must tell apart
  - the ideal function: a monolithic random oracle $\mathcal{RO}$
  - construction $\mathcal{S}[\mathcal{F}]$ calling an ideal primitive $\mathcal{F}$
- Express $\Pr(\text{success}|\mathcal{D})$ as a function of total cost of queries N
- Problem: in real world, $\mathcal{F}$ is available to adversary

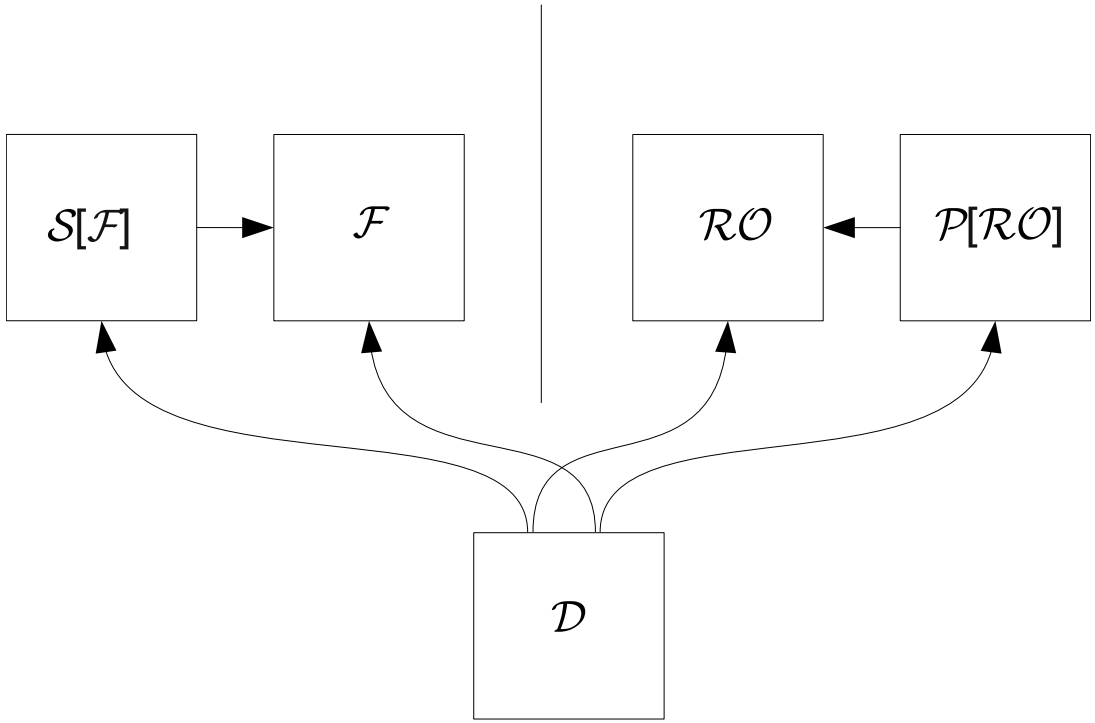
Generic security: indifferentiability [Maurer et al. (2004)]



Applied to hash functions in [Coron et al. (2005)]

- distinguishing mode-of-use from ideal function ($\mathcal{RO}$)
- covers adversary with access to primitive $\mathcal{F}$ at left
- additional interface, covered by a *simulator* at right

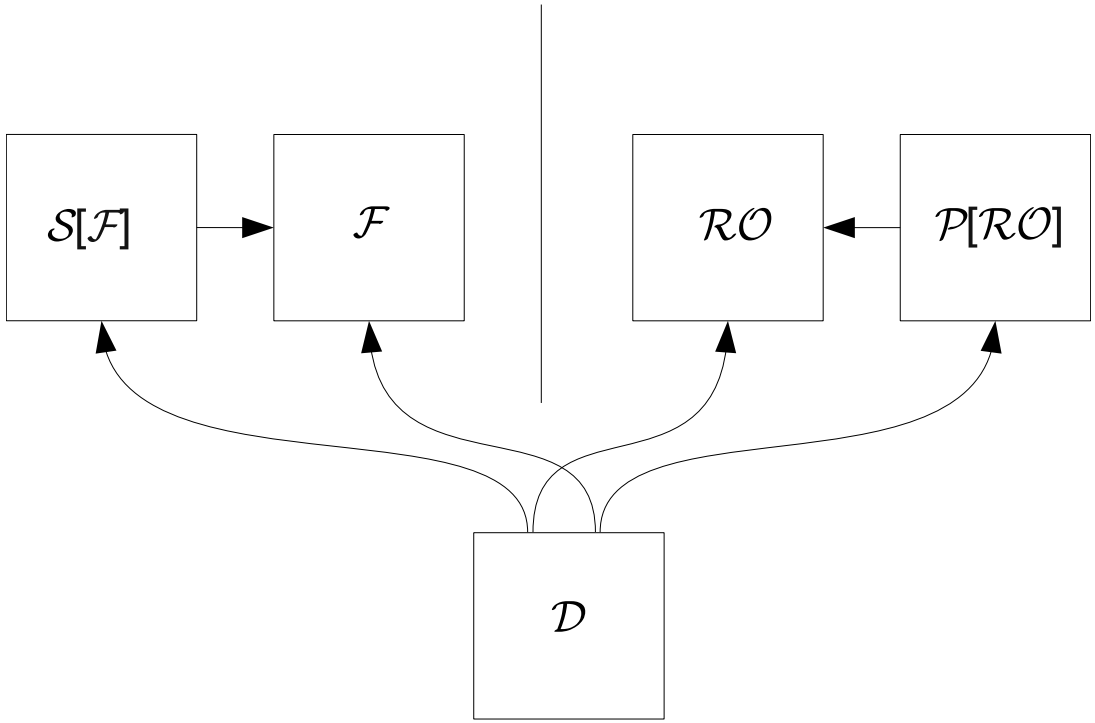
Generic security: indifferentiability [Maurer et al. (2004)]



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: $\mathcal{P}[\mathcal{RO}]$

Generic security: indifferentiability [Maurer et al. (2004)]



$$\mathsf{Adv}(q) = \left|\Pr\left(\mathcal{D}^{\mathcal{S}[\mathcal{F}],\mathcal{F}}\right) - \Pr\left(\mathcal{D}^{\mathcal{RO},\mathcal{P}[\mathcal{RO}]}\right)\right| \leq \epsilon(q)$$

Consequences of indifferentiability

- Let $\mathcal{D}$: n-bit output pre-image attack. Success probability:
  - for random oracle: $P_{\text{pre}}(\mathcal{D}|\mathcal{RO}) = q2^{-n}$
  - for our construction: $P_{\text{pre}}(\mathcal{D}|\mathcal{S}[\mathcal{F}]) = ?$
- A distinguisher $\mathcal{D}$ with $\text{Adv}(q) = P_{\text{pre}}(\mathcal{D}|\mathcal{S}[\mathcal{F}]) - P_{\text{pre}}(\mathcal{D}|\mathcal{RO})$
  - do pre-image attack
  - if success, conclude our construction; otherwise, $\mathcal{RO}$
- But we have a proven bound $\text{Adv}(q) \leq \epsilon(q)$, so

$$P_{\text{pre}}(\mathcal{D}|\mathcal{S}[\mathcal{F}]) \leq P_{\text{pre}}(\mathcal{D}|\mathcal{RO}) + \epsilon(q)$$

- Can be generalized to any attack

Consequences of indifferentiability

Theorem 2. *Let $\mathcal{H}$ be a hash function, built on underlying primitive π, and RO be a random oracle, where $\mathcal{H}$ and RO have the same domain and range space. Denote by $\boldsymbol{Adv}^{\mathrm{pro}}_{\mathcal{H}}(q)$ the advantage of distinguishing $(\mathcal{H}, \pi)$ from (RO, S), for some simulator S, maximized over all distinguishers $\mathcal{D}$ making at most q queries. Let* atk *be a security property of $\mathcal{H}$. Denote by $\boldsymbol{Adv}^{\mathrm{atk}}_{\mathcal{H}}(q)$ the advantage of breaking $\mathcal{H}$ under* atk*, maximized over all adversaries $\mathcal{A}$ making at most q queries. Then:*

$$\boldsymbol{Adv}^{\mathrm{atk}}_{\mathcal{H}}(q) \leq \boldsymbol{Pr}^{\mathrm{atk}}_{RO}(q) + \boldsymbol{Adv}^{\mathrm{pro}}_{\mathcal{H}}(q), \tag{1}$$

where $\boldsymbol{Pr}^{\mathrm{atk}}_{RO}(q)$ denotes the success probability of a generic attack against $\mathcal{H}$ under atk*, after at most q queries.*

[Andreeva, Mennink, Preneel, ISC 2010]

Limitations of indifferentiability

- Only about the mode
  - No security proof with a concrete primitive
- Only about single-stage games [Ristenpart et al., Eurocrypt 2011]
  - Example: hash-based storage auditing

$$Z = h(\mathsf{File} \| C)$$

Limitations of indifferentiability

- Only about the mode
  - No security proof with a concrete primitive
- Only about single-stage games [Ristenpart et al., Eurocrypt 2011]
  - Example: hash-based storage auditing

$$Z = h(\mathsf{File} \| C)$$

Outline

Symmetric crypto: what textbooks and intro's say

Symmetric cryptography primitives:

- Block ciphers
- Key stream generators
- Hash functions

And their modes-of-use

Picture by GlasgowAmateur

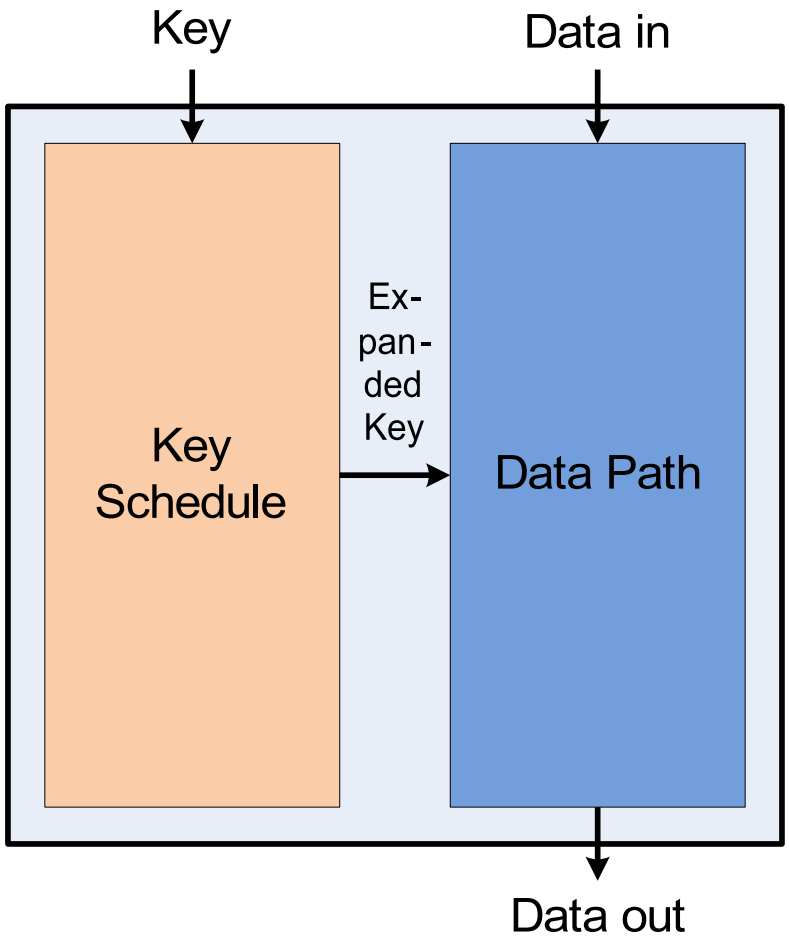
The truth about symmetric crypto today

Block ciphers:

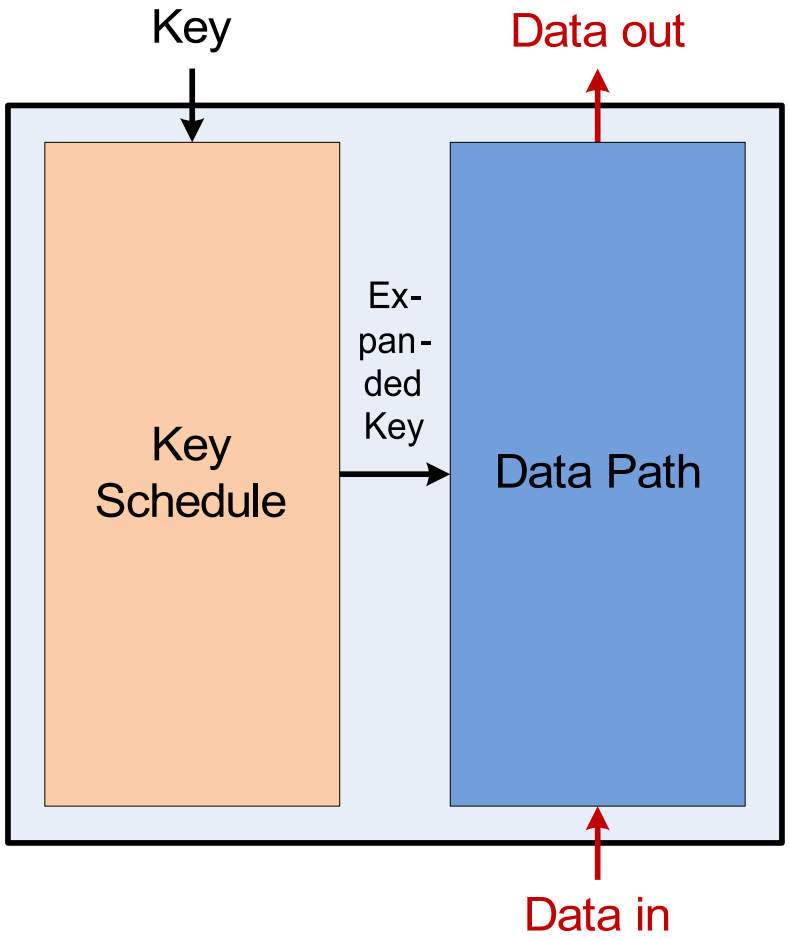
What block cipher are used for

- Hashing (Davies-Meyer) and its modes HMAC, MGF1, ...
- Block encryption: ECB, CBC, ...
- Stream encryption:
  - synchronous: counter mode, OFB, ...
  - self-synchronizing: CFB
- MAC computation: CBC-MAC, C-MAC, ...
- Authenticated encryption: OCB, GCM, CCM ...

Block cipher operation



Block cipher operation: the inverse



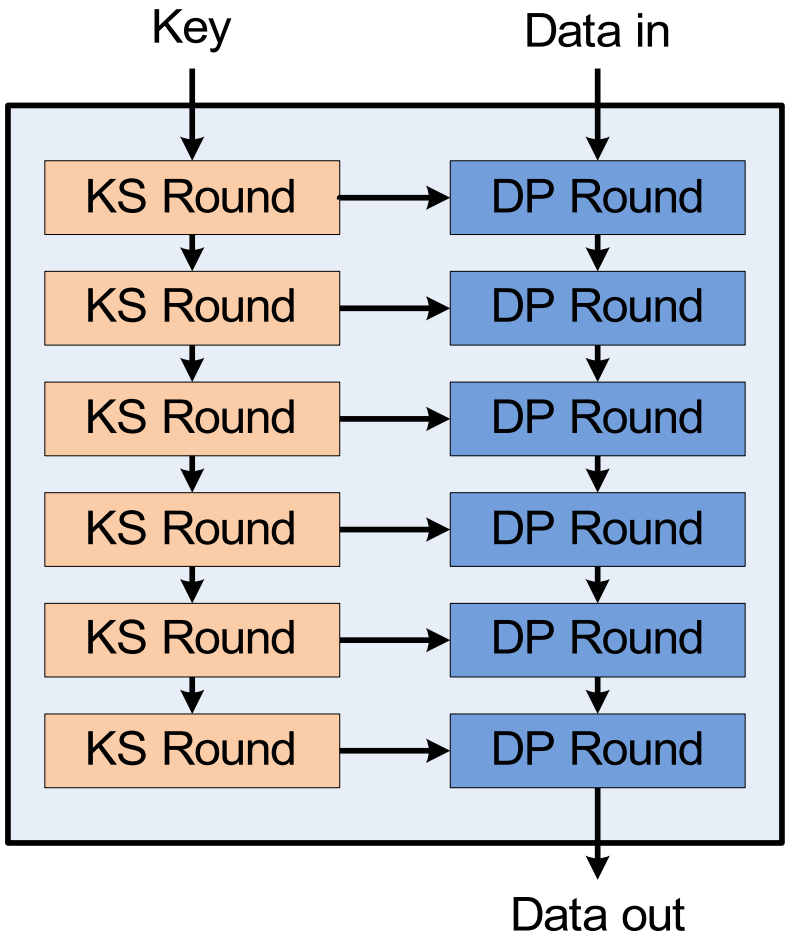
When do you need the inverse?

Indicated in red:

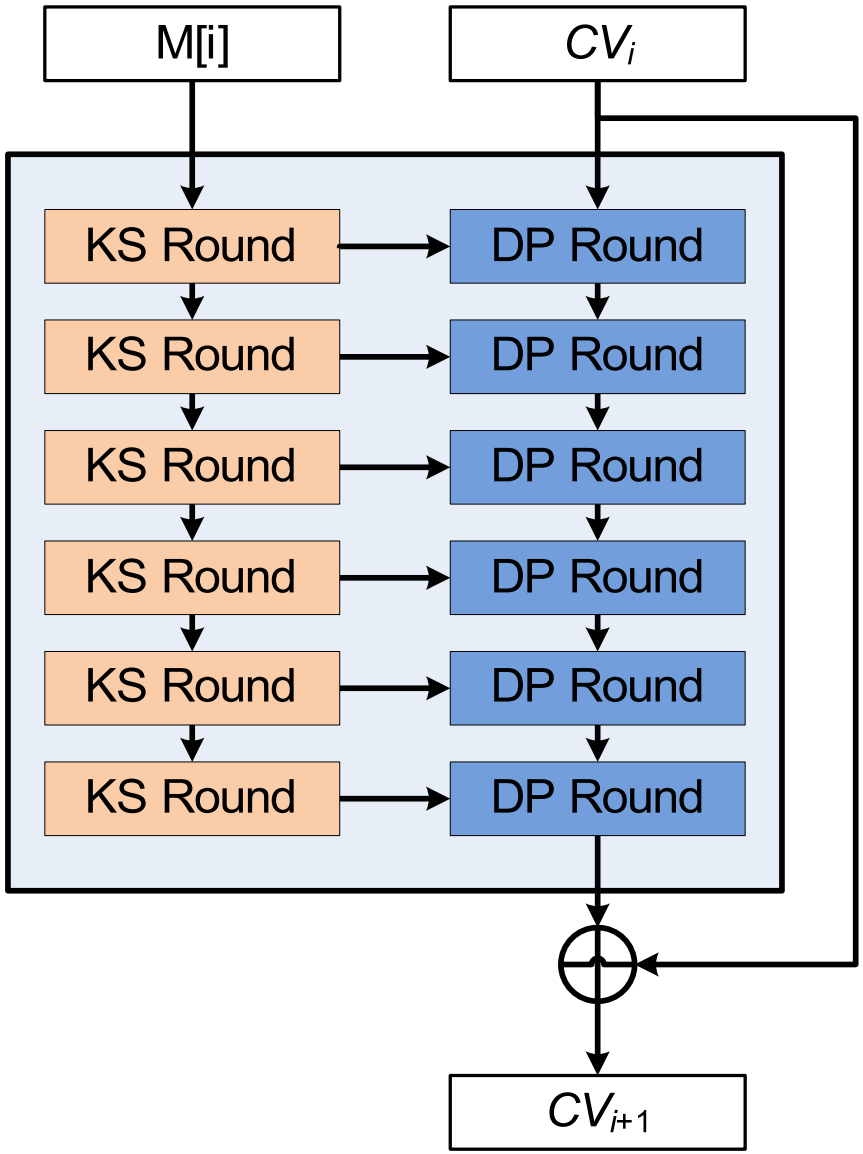
- Hashing and its modes HMAC, MGF1, ...
- Block encryption: ECB, CBC, ...
- Stream encryption:
  - synchronous: counter mode, OFB, ...
  - self-synchronizing: CFB
- MAC computation: CBC-MAC, C-MAC, ...
- Authenticated encryption: OCB, GCM, CCM ...
  - Most schemes with misuse-resistant claims

So for most uses you don't need the inverse!

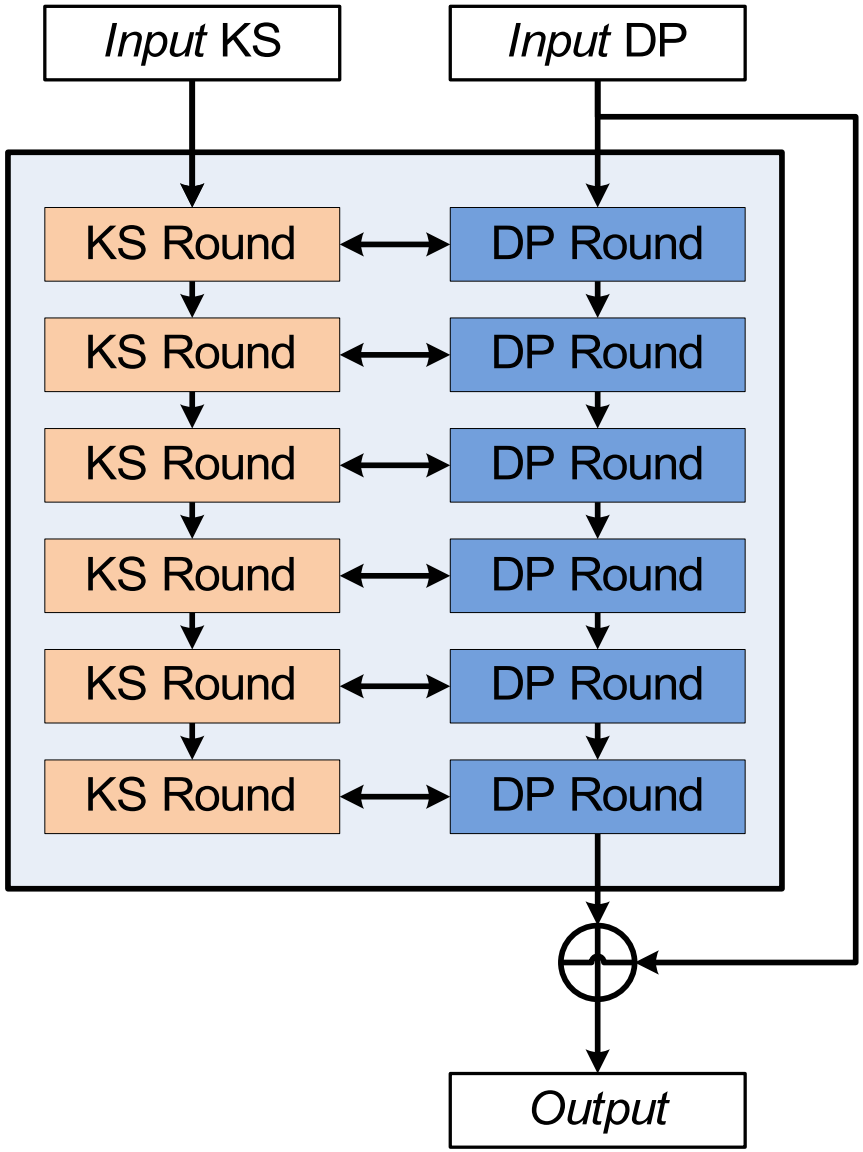
Block cipher internals



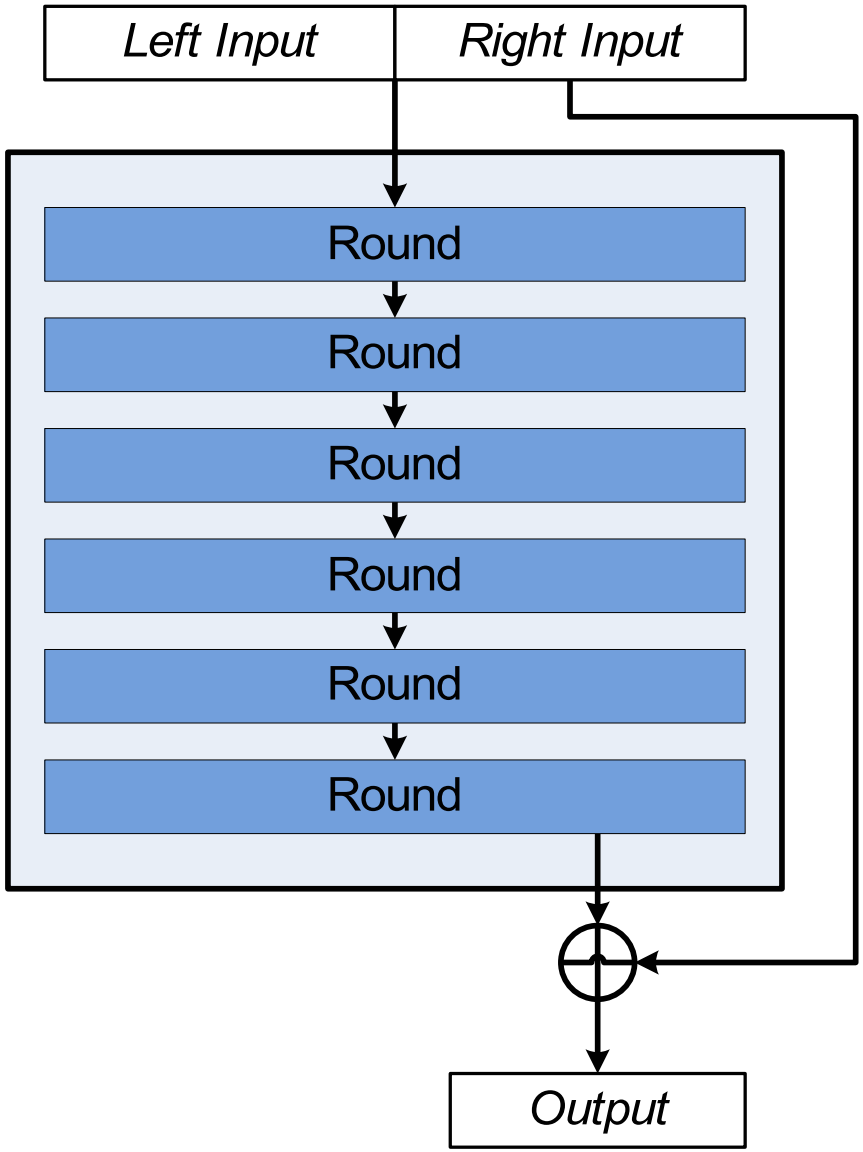
Davies-Meyer compression function



Removing restrictions not required in hashing



Simplifying the view: iterated permutation



Designing a permutation

- Remaining problem: design of iterated permutation
  - round function: good approaches known
  - asymmetry: round constants
- Advantages with respect to block ciphers:
  - less barriers $\Rightarrow$ more diffusion
  - no more need for efficient inverse
  - no more worries about key schedule

Examples of permutations

- In Salsa, Chacha, Spongent, Quark, Photon...
- In SHA-3 candidates: CubeHash, Grøstl, JH, MD6, ...
- In CAESAR candidates: Ascon, Icepole, Norx, π-cipher, Primates, Stribob, ...
- In recent proposals: Gimli, Xoodoo

And of course in KECCAK

What textbooks and intro's should say

Symmetric cryptography primitives:

- Block ciphers
- Key stream generators
- **Permutations**

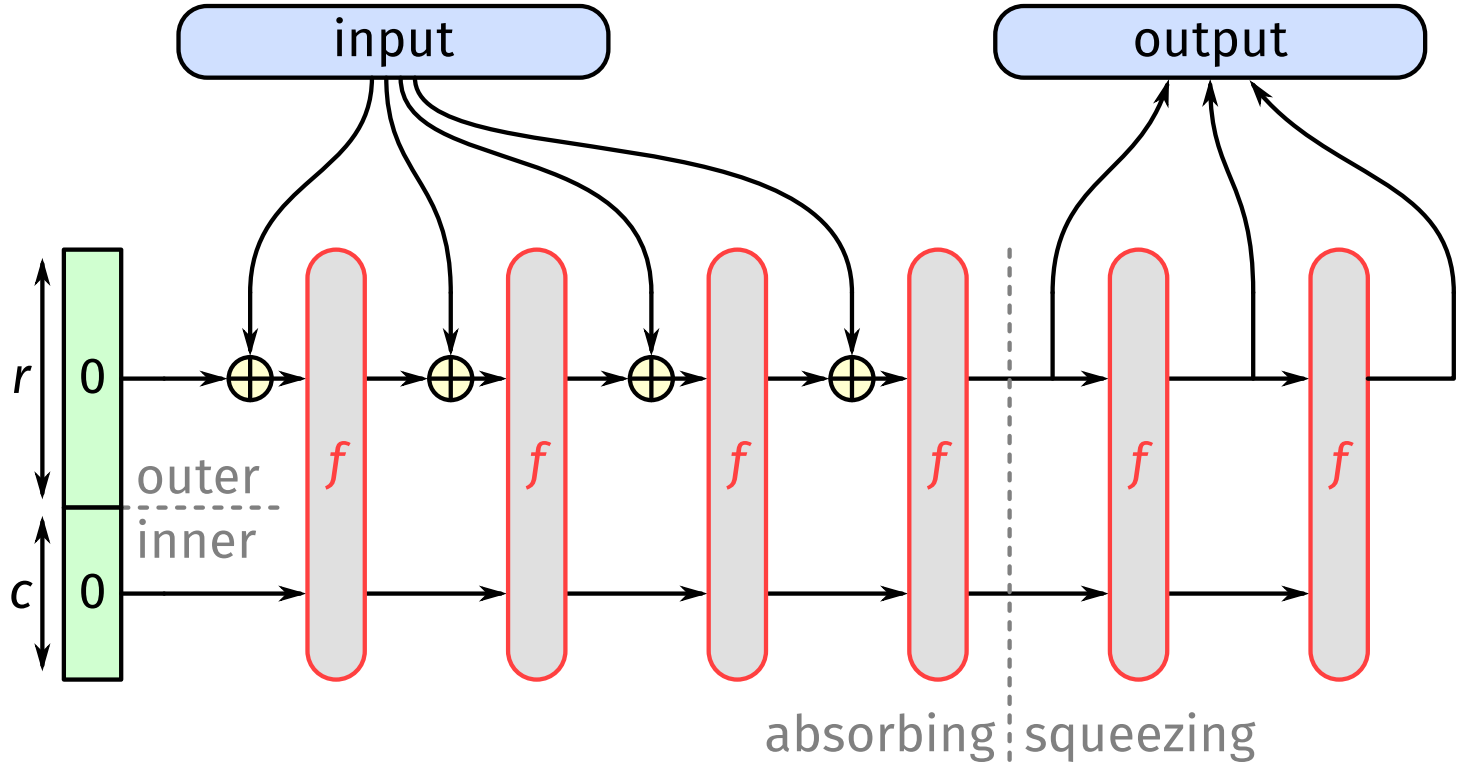
And their modes-of-use

Picture by Sébastien Wiertz

Outline

Outline

The sponge construction



- Calls a b-bit permutation f, with $b = r + c$
  - r bits of *rate*
  - c bits of *capacity* (security parameter)
- Natively implements a XOF

Generic security of the sponge construction

Theorem (Bound on the $\mathcal{RO}$-differentiating advantage of sponge)

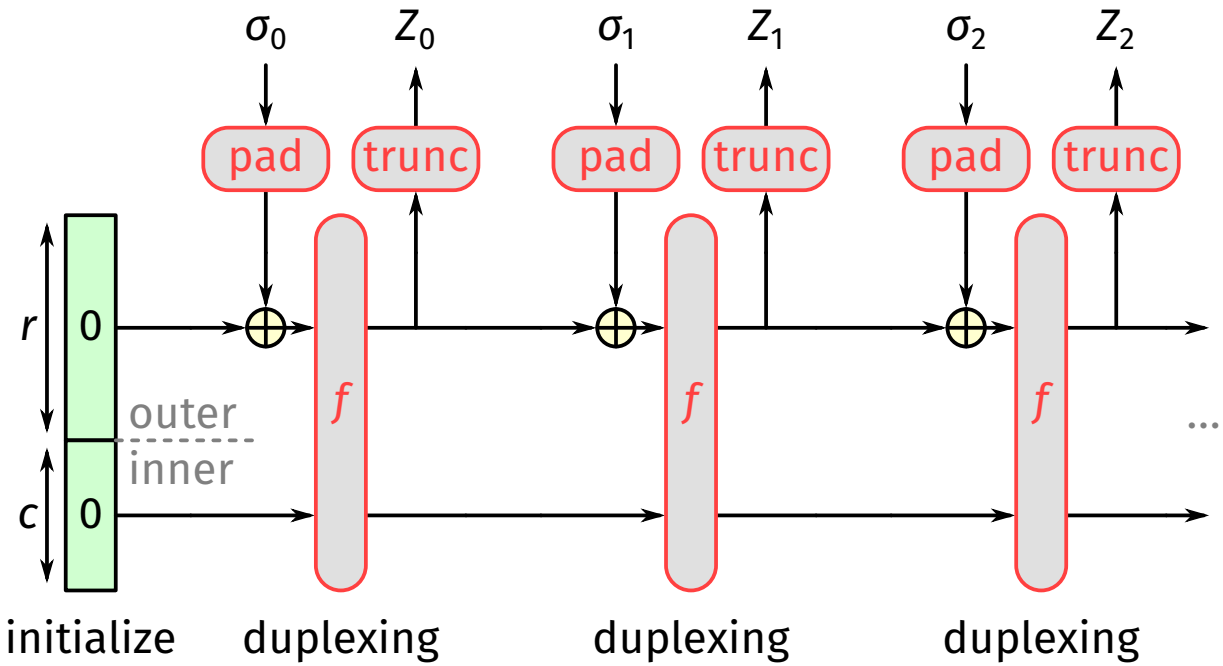
$$A \leq \frac{N^2}{2^{c+1}}$$

A: differentiating advantage of random sponge from random oracle
N: total data complexity *c: capacity* *[Keccak Team, Eurocrypt 2008]*

Preimage resistance	$2^{\min(n,c/2)}$
Second-preimage resistance	$2^{\min(n,c/2)}$
Collision resistance	$2^{\min(n/2,c/2)}$
Any other attack	$2^{\min(\mathcal{RO},c/2)}$

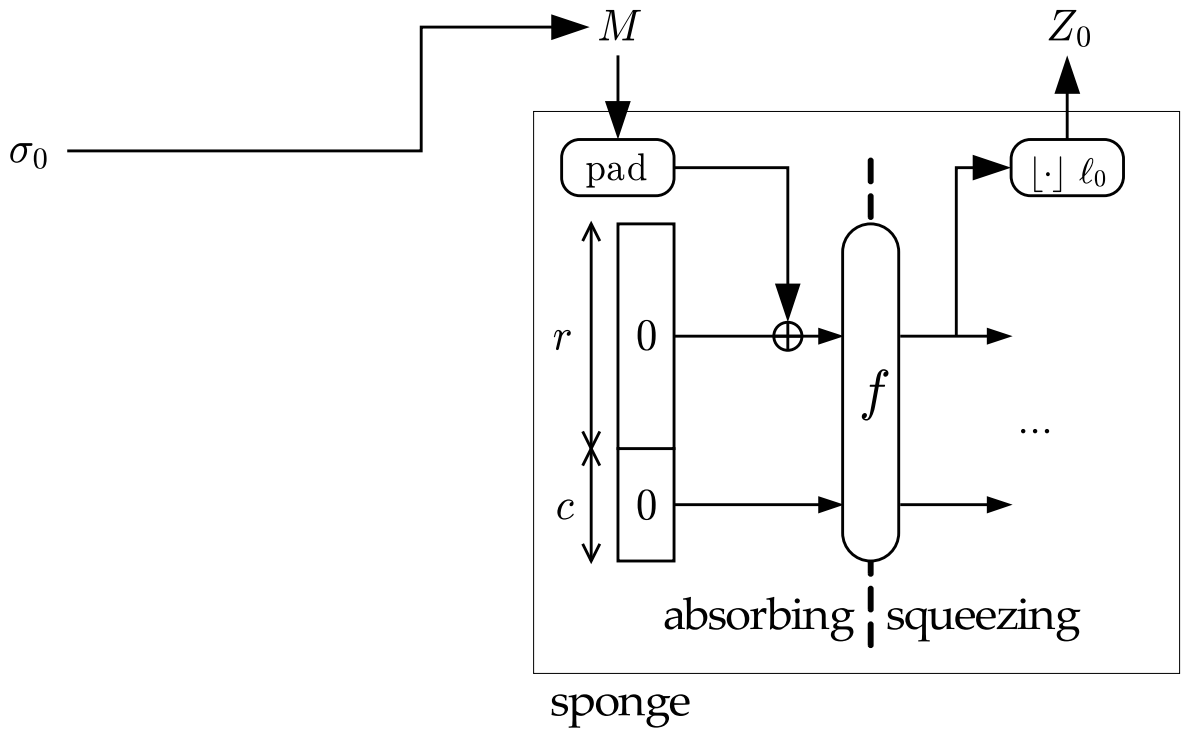
Outline

The duplex construction



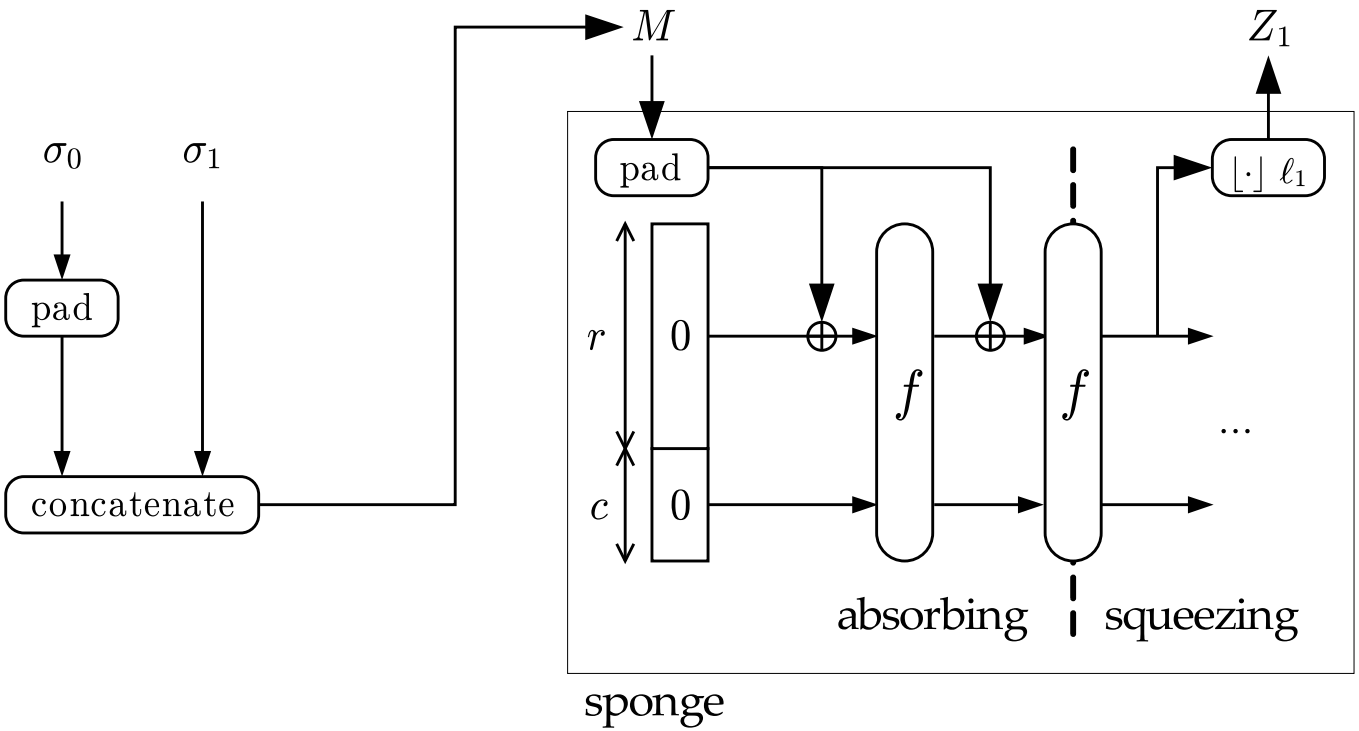
- Object: $D = \text{DUPLEX}[f, \text{pad}, r]$
- Requesting ℓ-bit output $Z = D.\text{duplexing}(\sigma, \ell)$
  - input σ and output Z limited in length
  - Z depends on all previous inputs

Generating duplex responses with a sponge



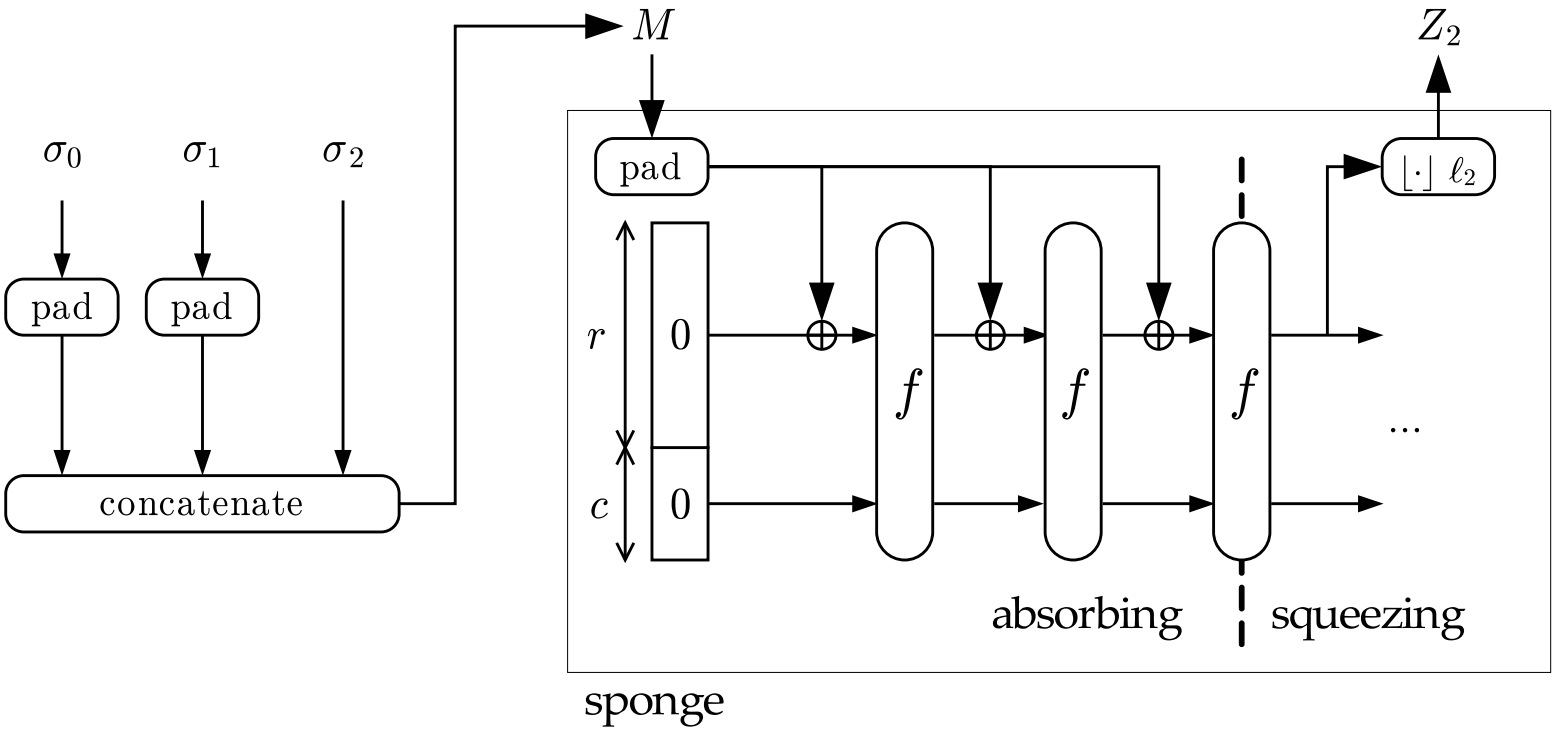
$$Z_0 = \mathsf{sponge}(\sigma_0, \ell_0)$$

Generating duplex responses with a sponge



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

Generating duplex responses with a sponge



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

Security of the duplex construction

Duplexing-sponge lemma

Every output block of a duplex object $\textsc{duplex}[f, \text{pad}, r]$ is a valid output of $\textsc{sponge}[f, \text{pad}, r]$

Proof is trivial

Corollary

The security of $\textsc{duplex}[f, \text{pad}, r]$ can be reduced to that of $\textsc{sponge}[f, \text{pad}, r]$

Security of the duplex construction

Duplexing-sponge lemma

Every output block of a duplex object $\textsc{duplex}[f, \text{pad}, r]$ is a valid output of $\textsc{sponge}[f, \text{pad}, r]$

Proof is trivial

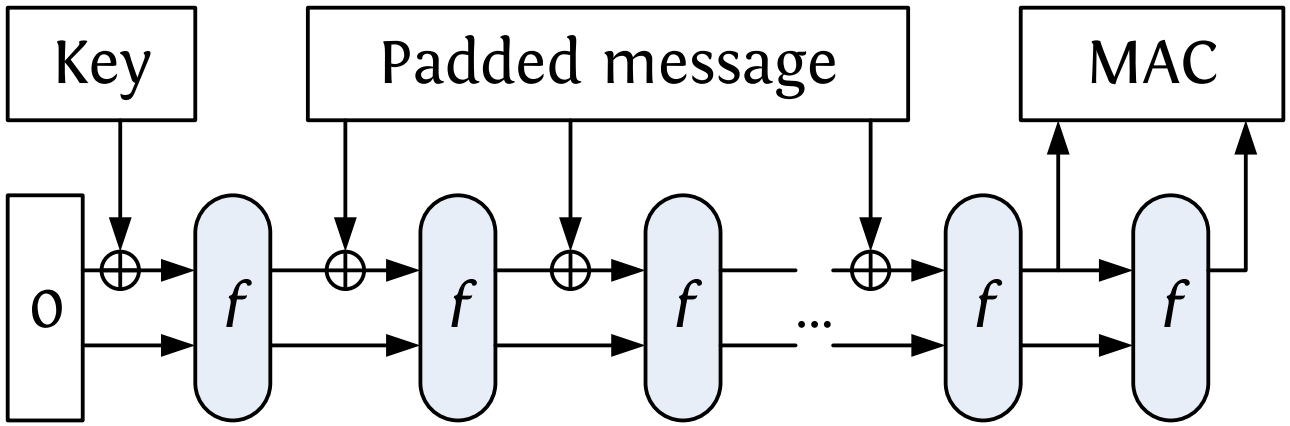
Corollary

The security of $\textsc{duplex}[f, \text{pad}, r]$ can be reduced to that of $\textsc{sponge}[f, \text{pad}, r]$

Outline

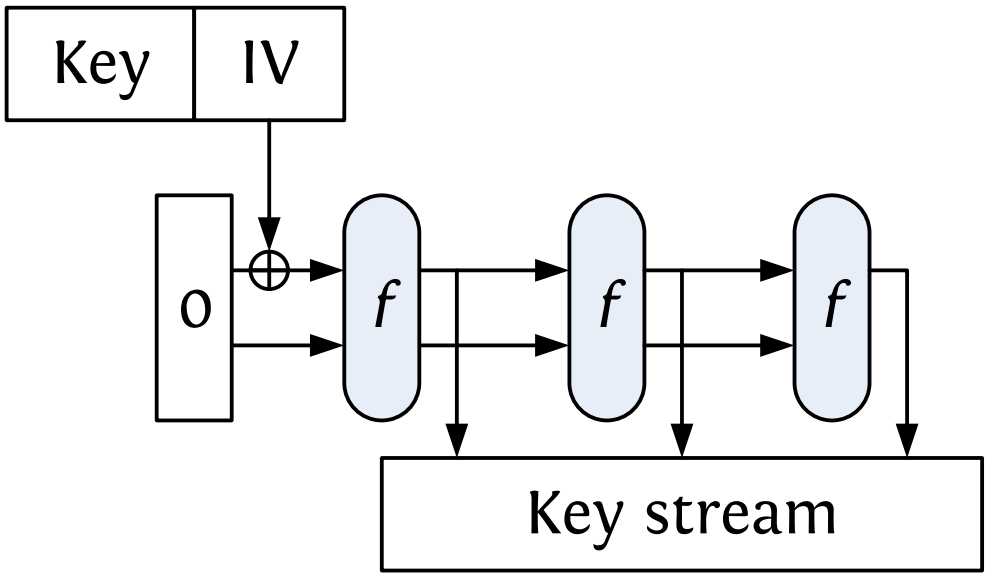
Outline

Message authentication codes



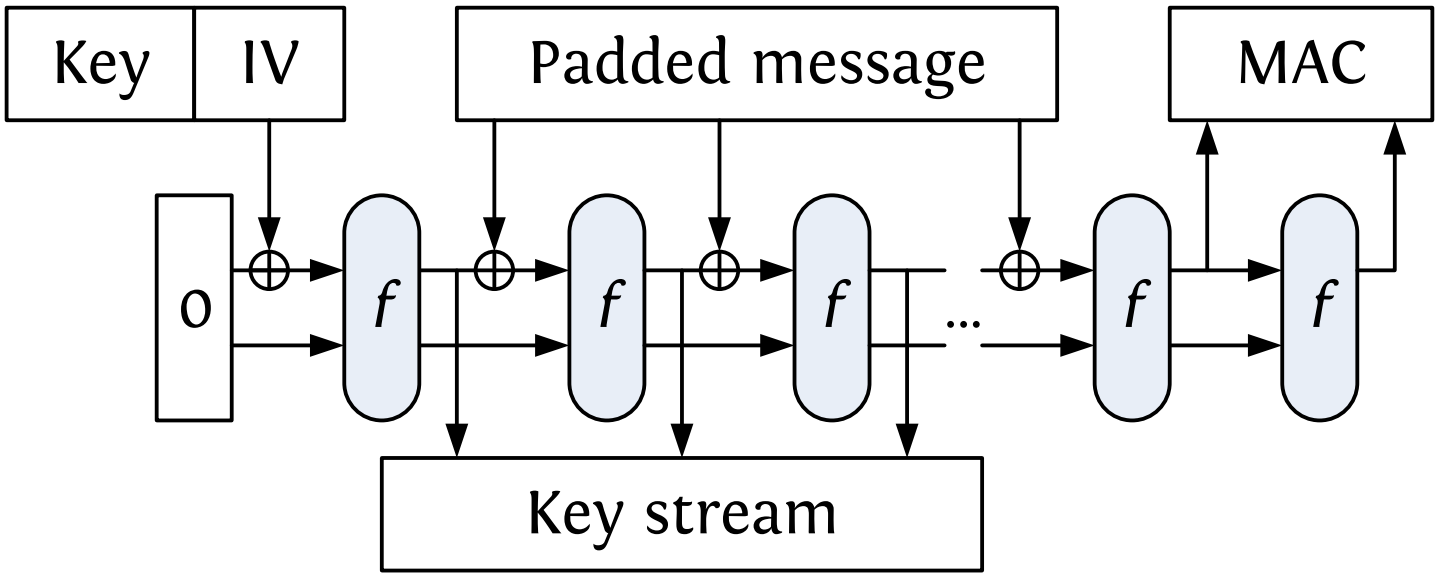
- Using sponge
- See also **KMAC** [NIST SP 800-185]

Stream encryption



- Using sponge
- Long output stream per IV: similar to OFB mode
- Short output stream per IV: similar to counter mode

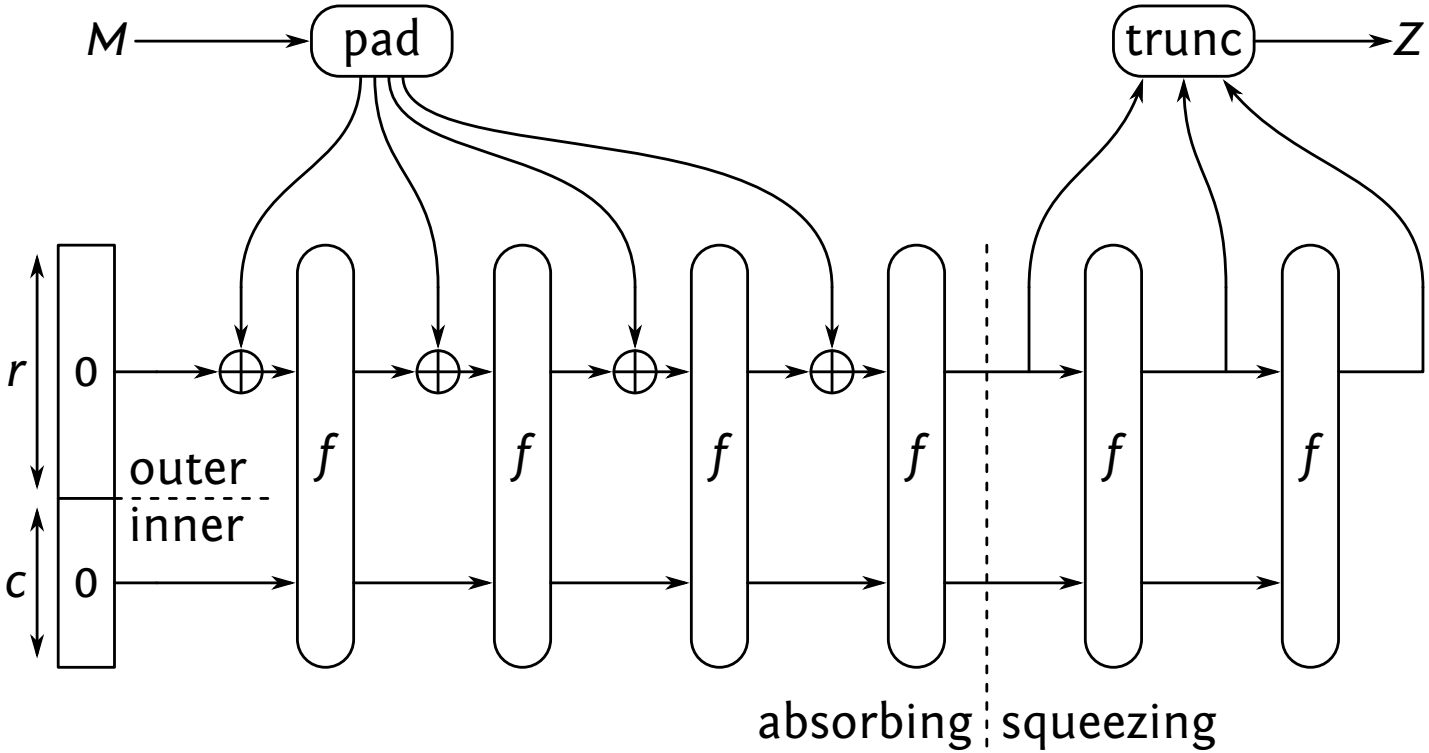
Authenticated encryption: spongeWrap



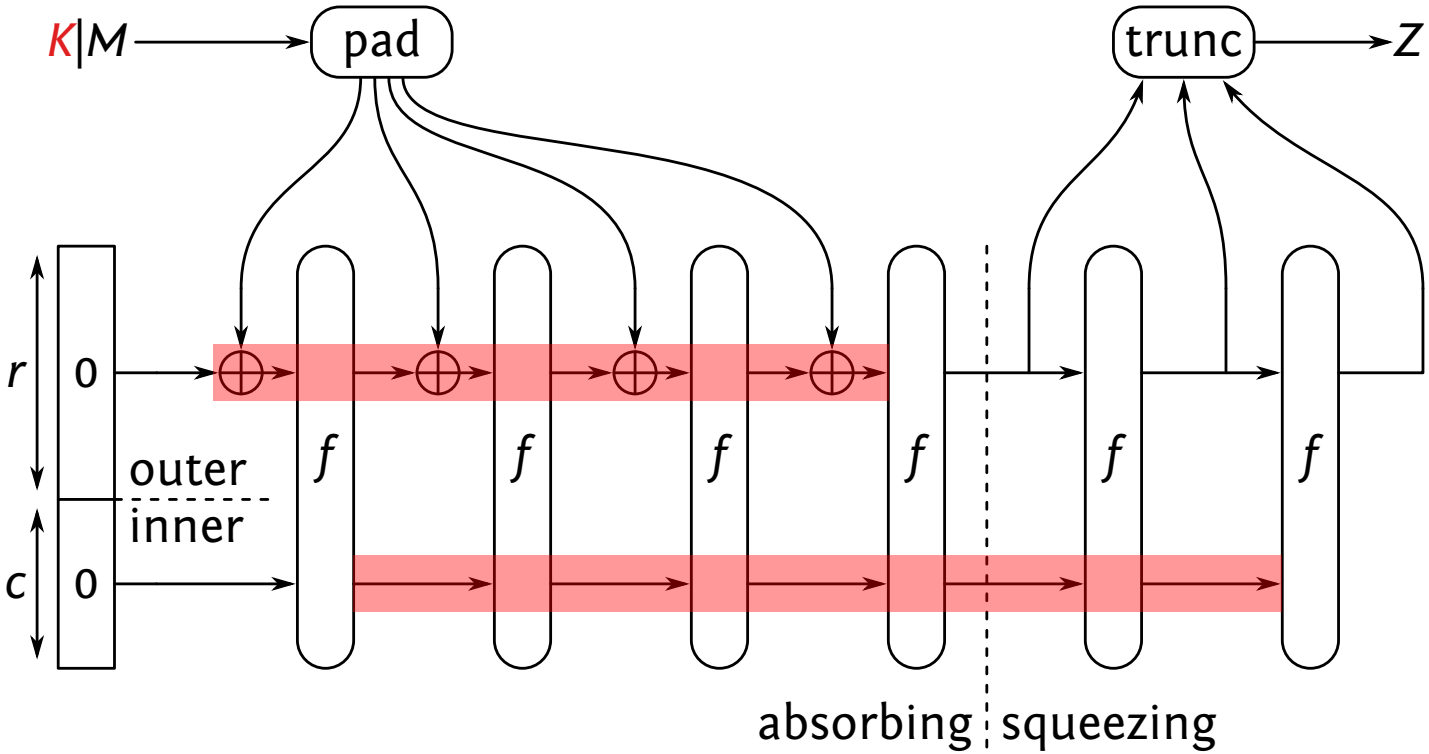
- Using duplex
- Adopted by several CAESAR and NIST LWC candidates

[Keccak Team, SAC 2011]

Outer keyed sponge

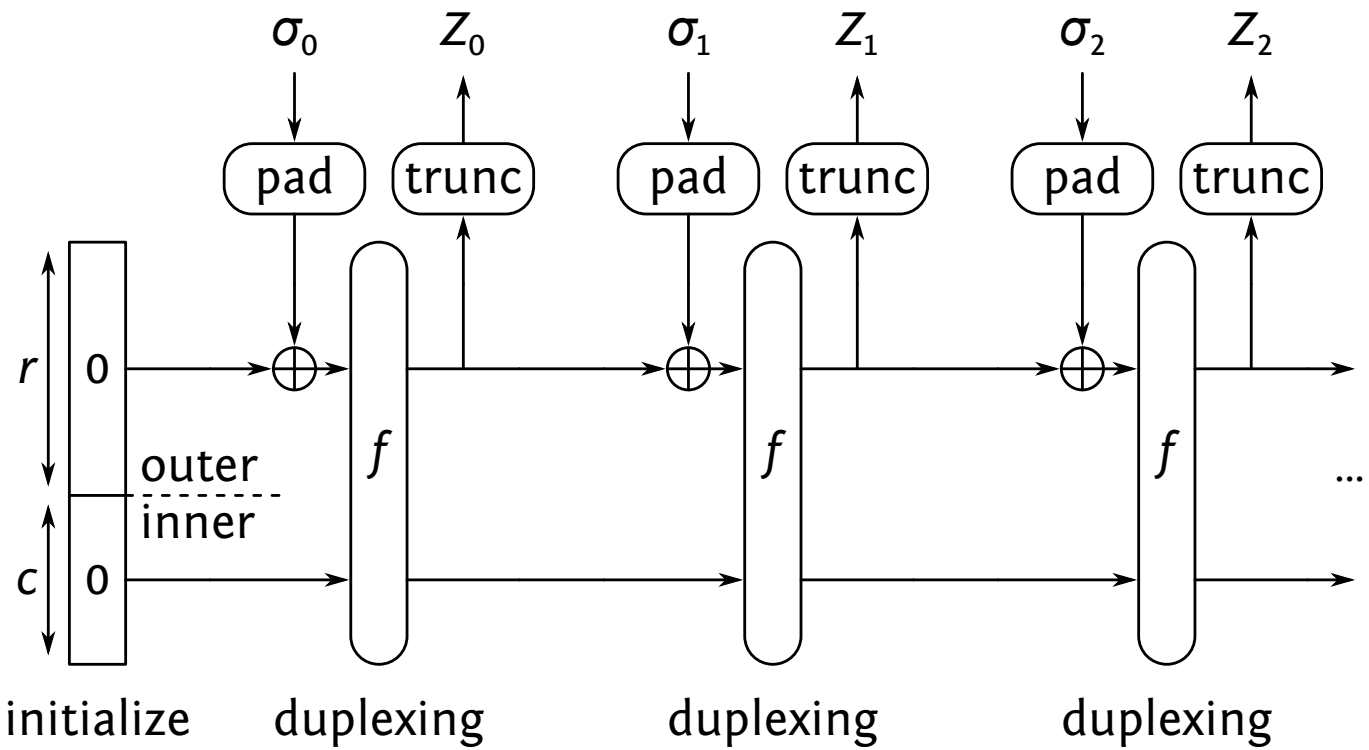


Outer keyed sponge



$$\mathrm{OKS}^f_K(M) = \mathrm{SPONGE}^f(K||M)$$

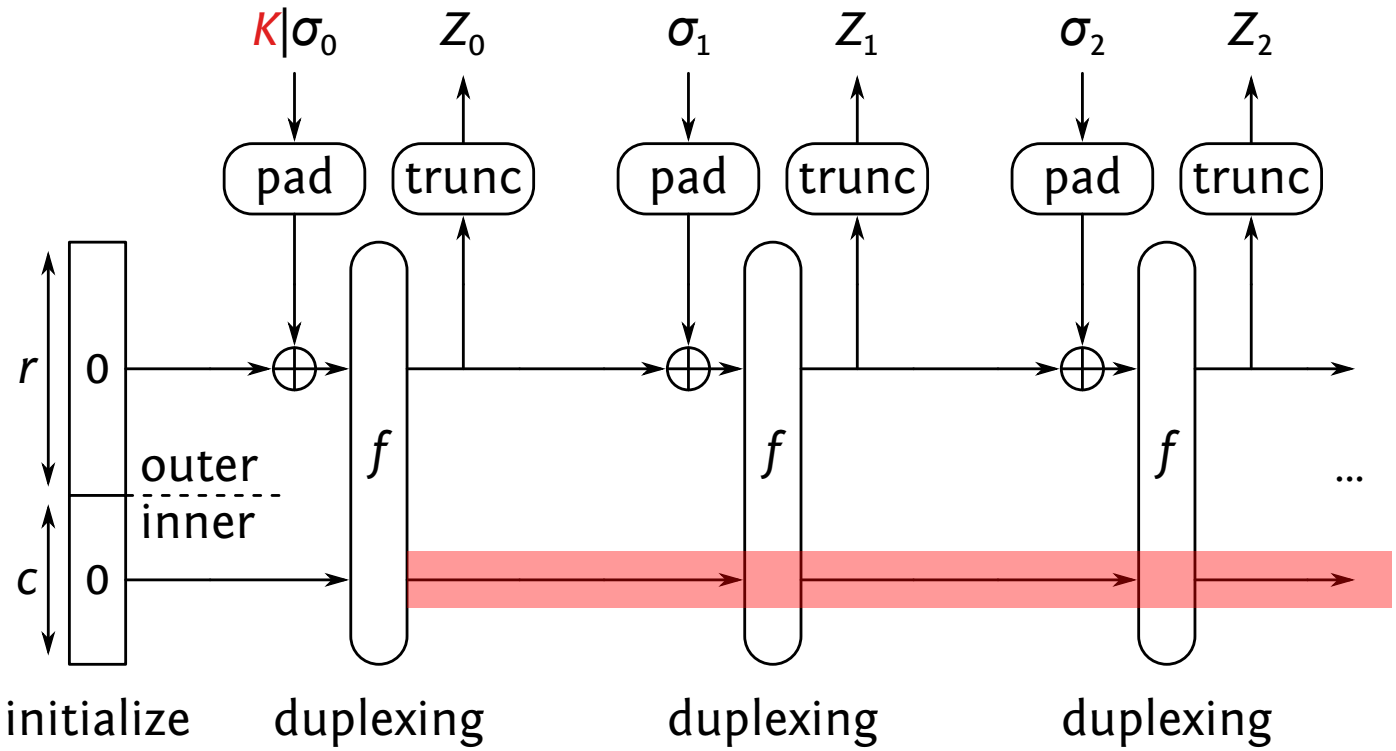
Outer keyed duplex



Duplexing-sponge lemma

$Z_i = \text{SPONGE}(\sigma_0||\text{pad}||\ldots||\sigma_i)$

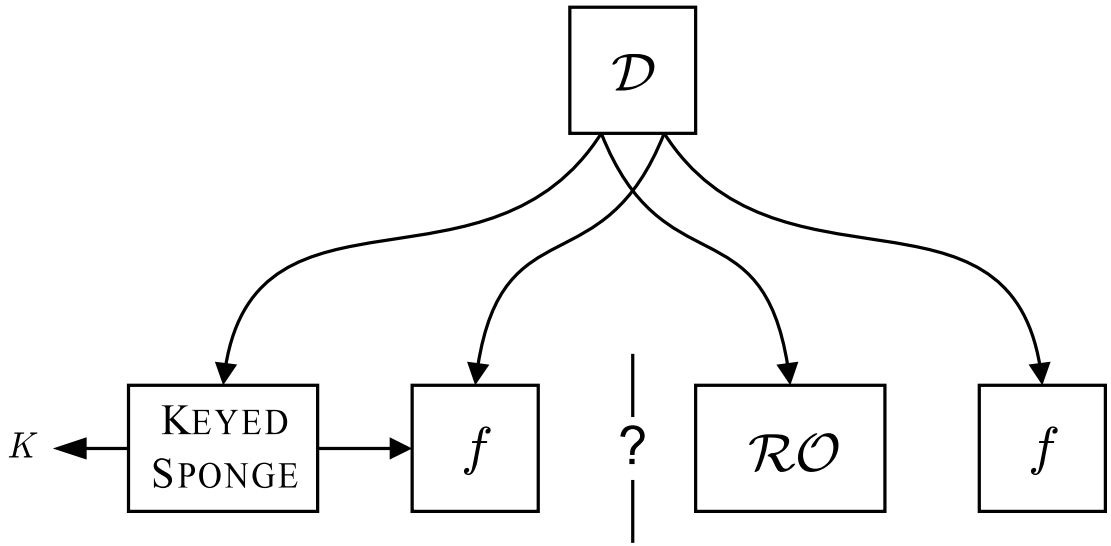
Outer keyed duplex



Duplexing-sponge lemma

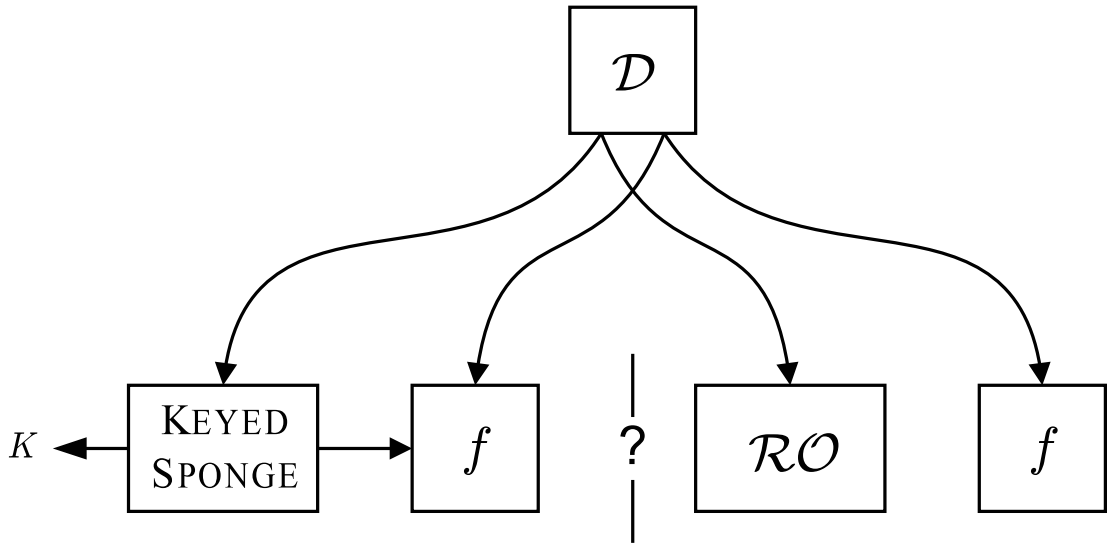
$Z_i = \text{SPONGE}(K||\sigma_0||\text{pad}||\ldots||\sigma_i) \Rightarrow$ equivalent to OKS_K

Keyed sponge: distinguishing setting



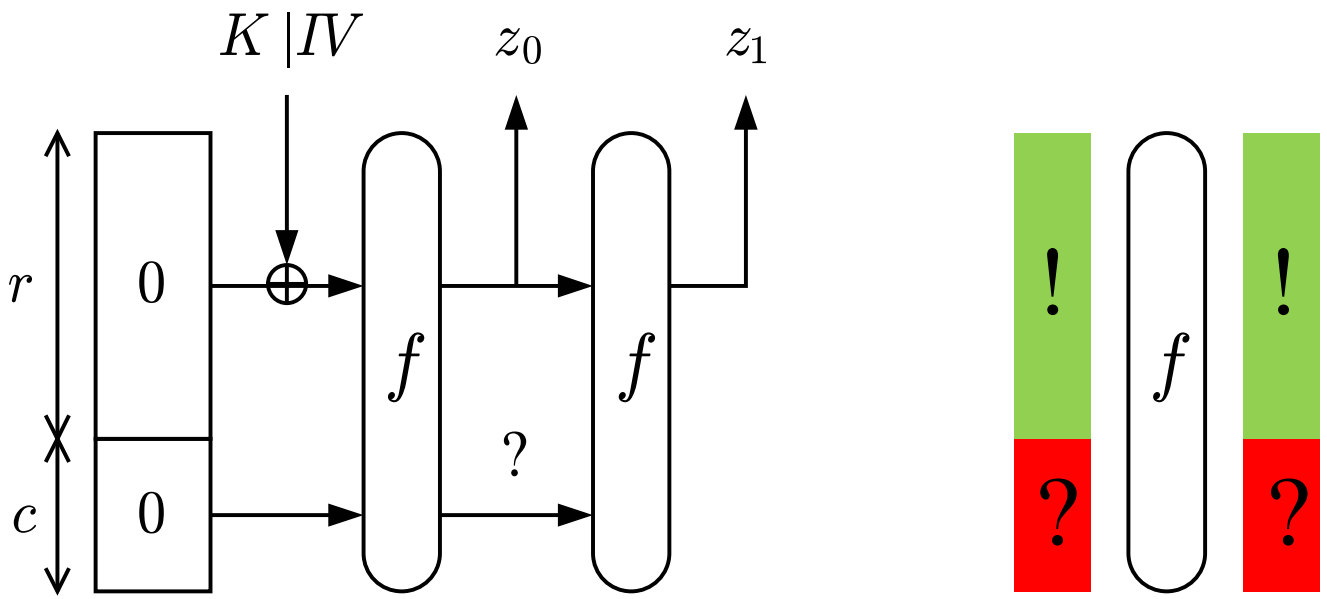
- Straightforward bound: $M^2/2^{c+1} + M/2^k$
- Security strength s: expected complexity of succesful attack
  - strength s means attack complexity 2^s
  - bounds can be converted to security strength statements
- Here: $s \leq \min(c/2, k)$
  - e.g., $s = 128$ requires $c = 256$ and $k = 128$
  - $c/2$: birthday bound

More fine-grained attack complexity



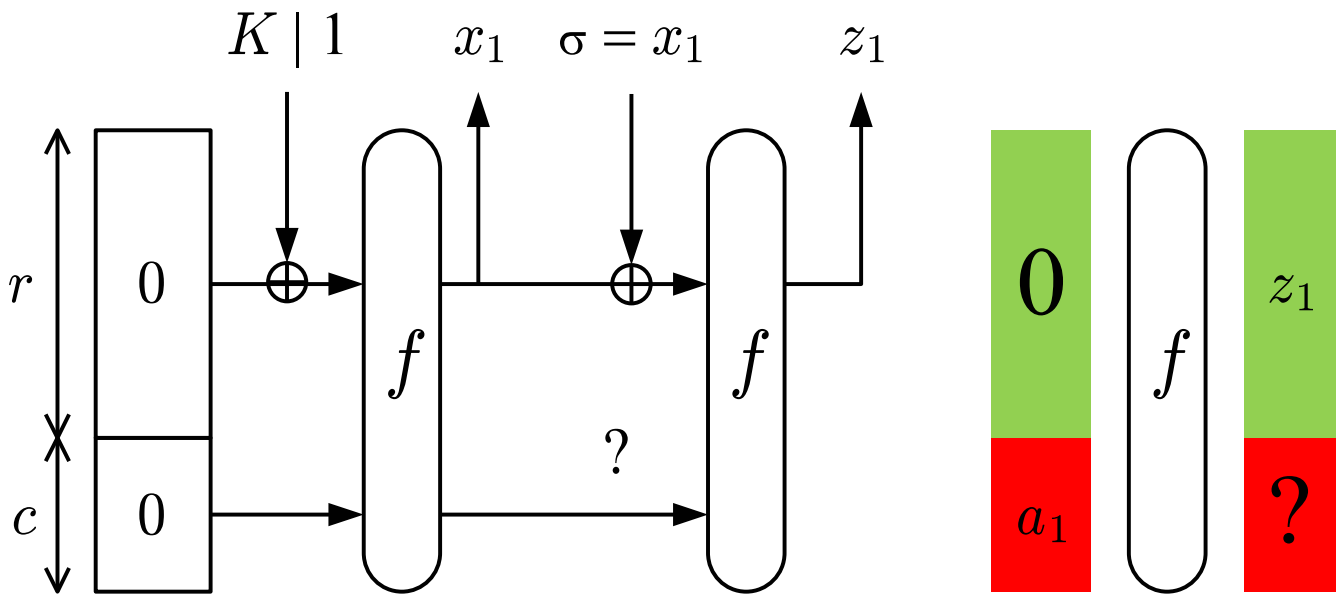
- Splitting attack complexity:
  - queries to construction: data complexity M
  - queries to f or f^{-1}: computational complexity N
- Our ambition around 2010: $M^2/2^{c+1} + NM/2^c + N/2^k$
- If we limit data complexity $M \leq 2^a \lll 2^{c/2}$:
  - $s \leq \min(c - a, k)$
  - e.g., $s = 128$ and $a = 64$ require $c = 192$ and $k = 128$

Intuition behind $NM/2^c$



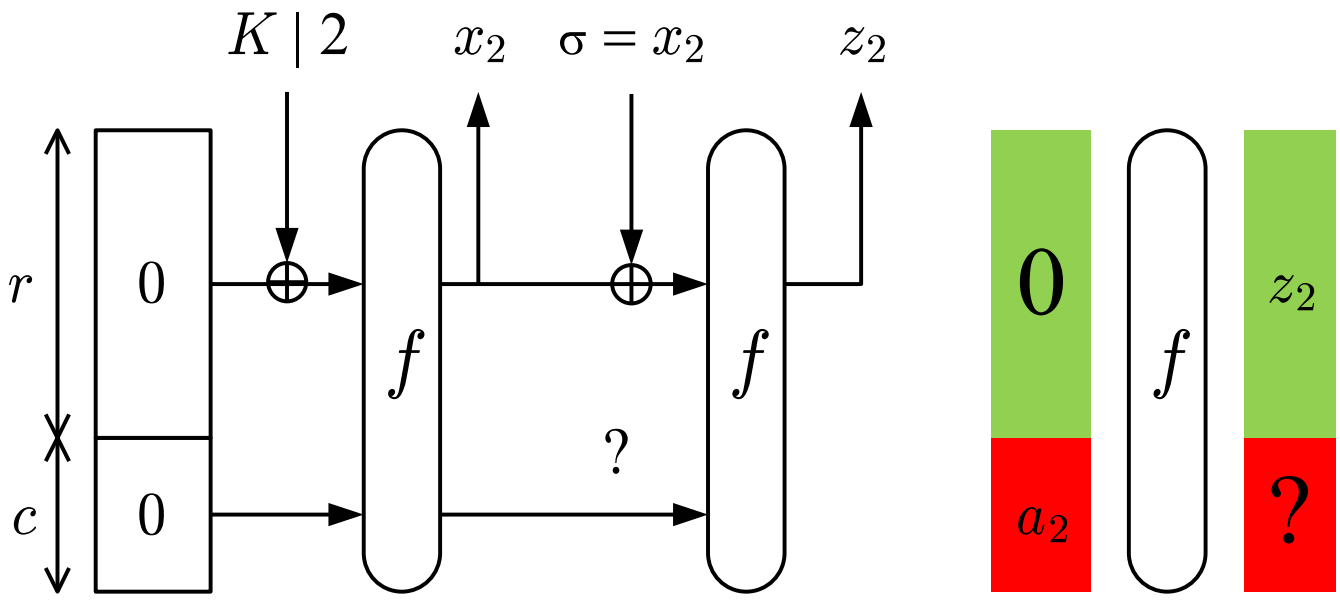
- Typically just one instance with the same partial r-bit input
- Success probability per guess: $1/2^c$

Intuition behind $NM/2^c$



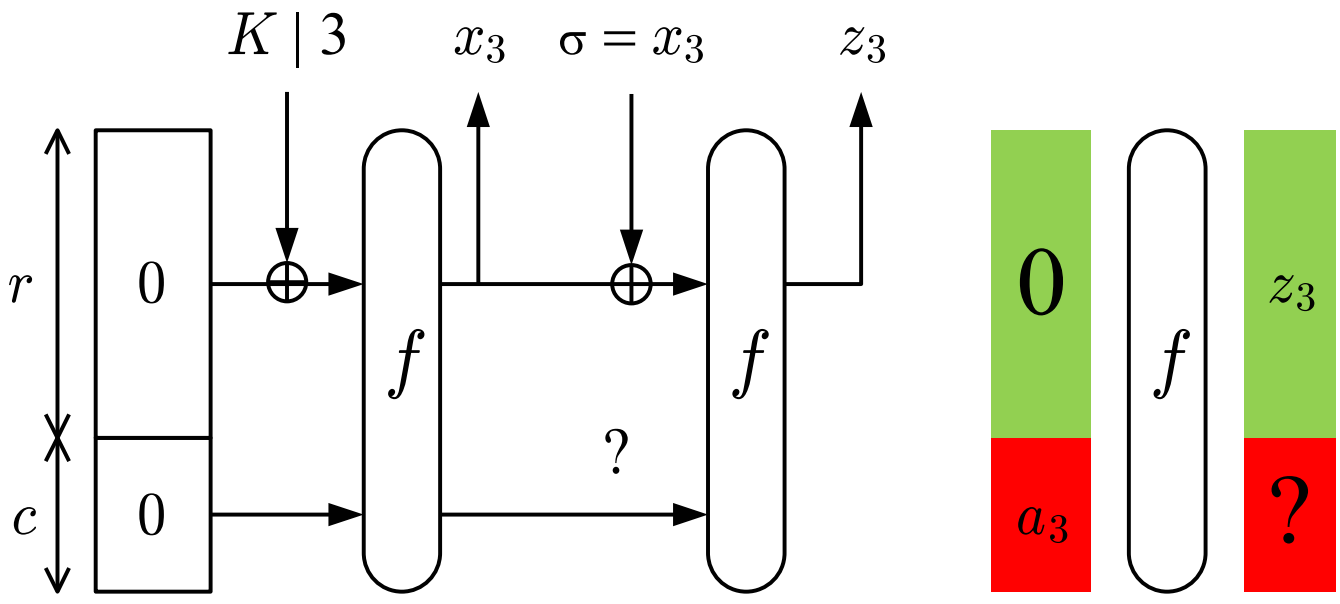
- Multiple instances ($\mu \leq M$) with same partial r-bit input
- Success probability per guess: $\mu/2^c$

Intuition behind $NM/2^c$



- Multiple instances ($\mu \leq M$) with same partial r-bit input
- Success probability per guess: $\mu/2^c$

Intuition behind $NM/2^c$

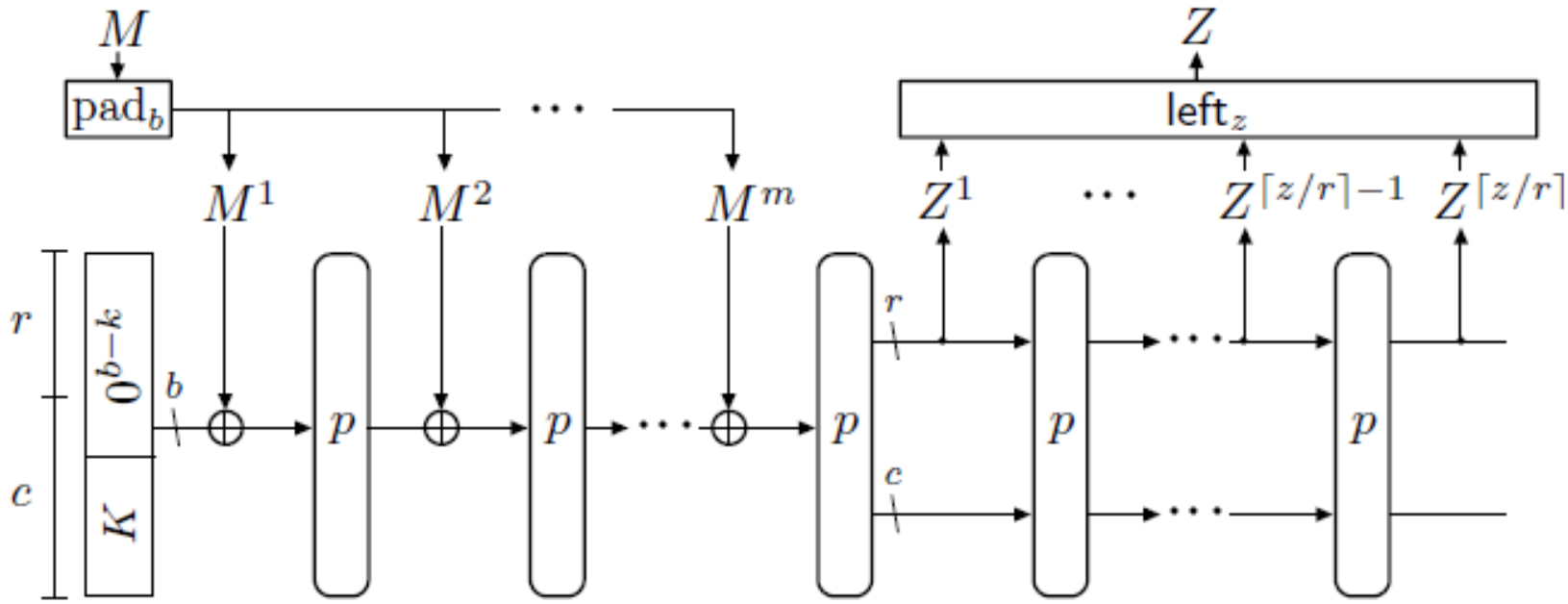


- Multiple instances ($\mu \leq M$) with same partial r-bit input
- Success probability per guess: $\mu/2^c$

Proof evolution

- Outer keyed sponge
  [Keccak Team, SKEW 2011]
- Inner keyed sponge
  [Chang, Dworkin, Hong, Kelsey, Nandi, 2012]
- Security beyond $2^{c/2}$
  [Jovanovic, Luykx, Mennink, Asiacrypt 2014]
- Inner and outer keyed sponges, multi-target
  [Andreeva, Daemen, Mennink, Van Assche, FSE 2015]
- Partially full-state sponge-based AE
  [Sasaki, Yasuda, CT-RSA 2015]
- Full-state keyed sponge (but fixed output size)
  [Gaži, Pietrzak, Tessaro, Crypto 2015]
- Full-state keyed sponge and duplex
  [Mennink, Reyhanitabar, Vizár, Asiacrypt 2015]
- Improved security of the outer keyed sponge
  [Naito, Yasuda, FSE 2016]

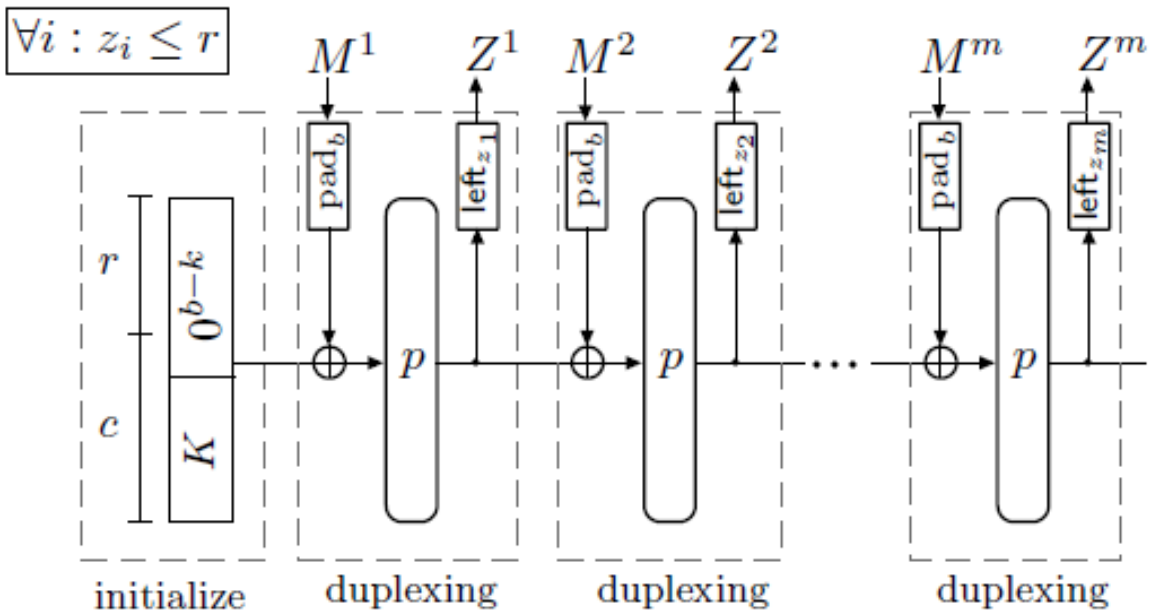
Full-state absorbing!



Absorbing on full permutation width does not degrade bounds

[Mennink, Reyhanitabar, Vizár, Asiacrypt 2015]

Full-state absorbing!

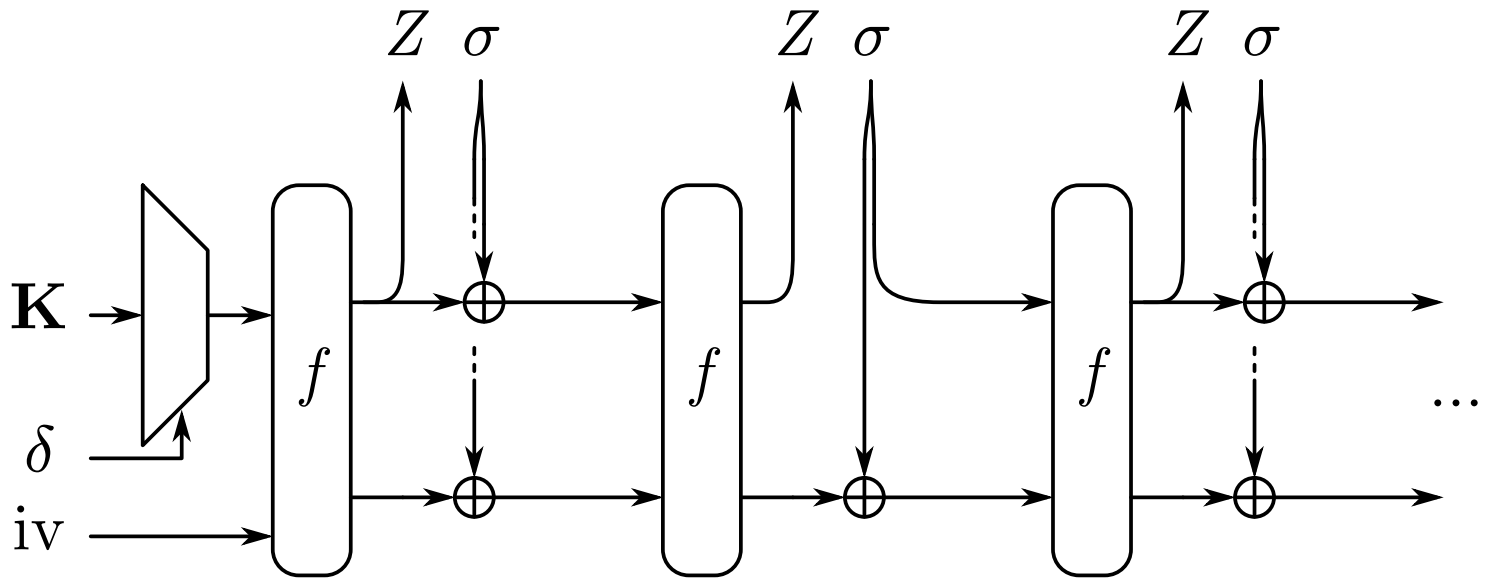


Absorbing on full permutation width does not degrade bounds

[Mennink, Reyhanitabar, Vizár, Asiacrypt 2015]

Outline

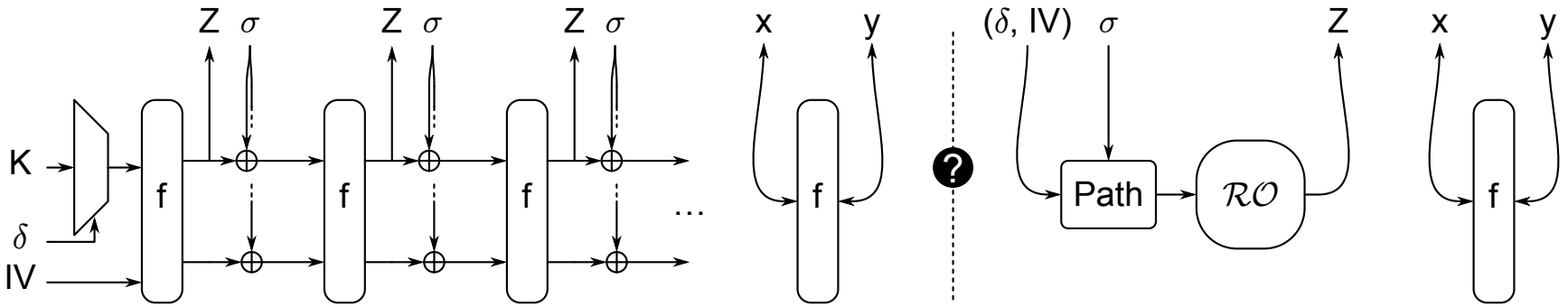
Keyed duplex



- Initial state: concatenation of key $k = \mathbf{K}[\delta]$ and IV
- Full-state absorbing, no padding: $|\sigma| = b$
- Re-phased: f, Z, σ instead of σ, f, Z

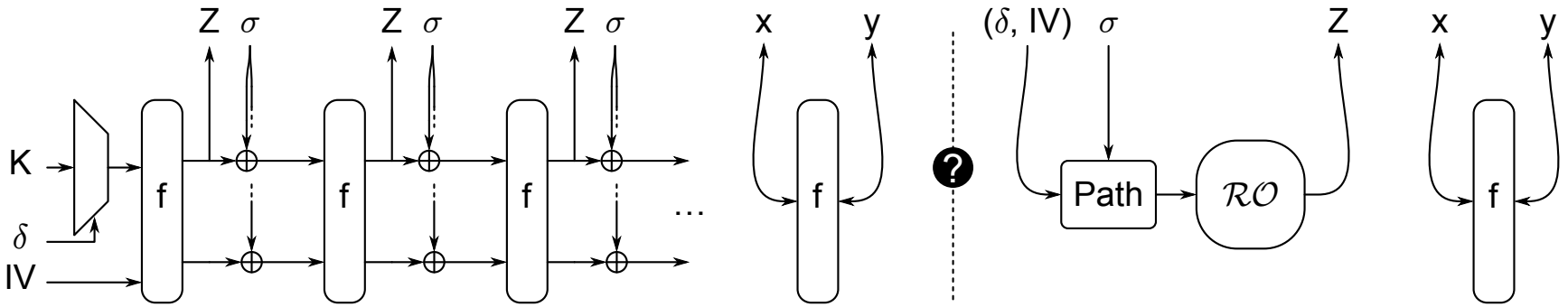
$\approx$ all keyed sponge functions are modes of this

Generic security of keyed duplex: the setup



- Ideal function: Ideal eXtendable Input Function (IXIF)
  - $\mathcal{RO}$-based object with duplex interface
  - Independent outputs Z for different paths
- Further refine adversary's capability
  - L: # queries to keyed duplex/$\mathcal{RO}$ with repeated path
  - q_{IV} : $\max_{IV}$ # init queries with different keys

Generic security of keyed duplex: the bound



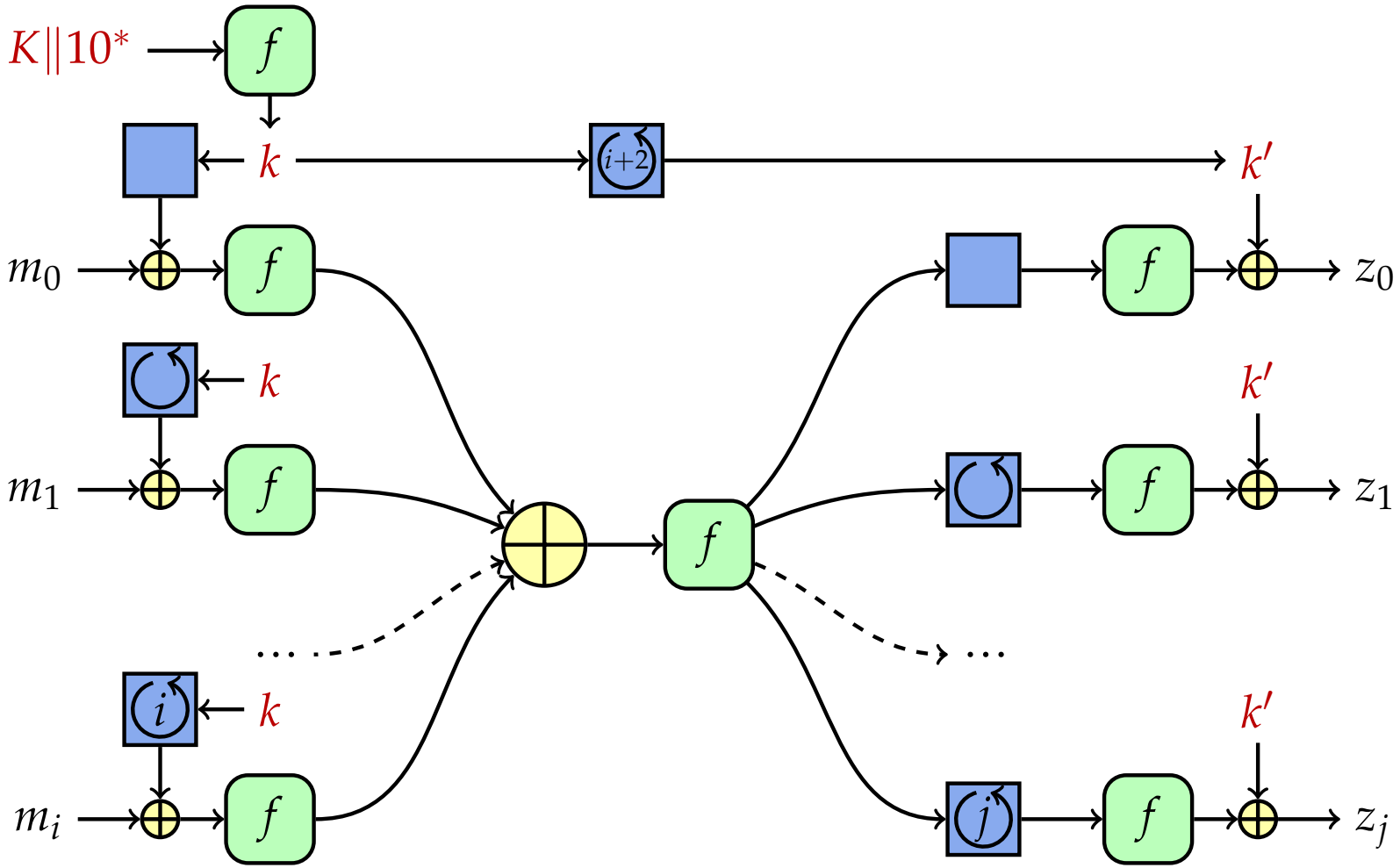
$$L^2/2^{c+1} + (L + 2\nu)N/2^c + q_{\mathsf{IV}}N/2^k + M^2/2^b + \ldots$$

with ν: chosen such that probability of ν-wise multi-collision in set of M r-bit values is negligible

[Daemen, Mennink, VA, Asiacrypt 2017]

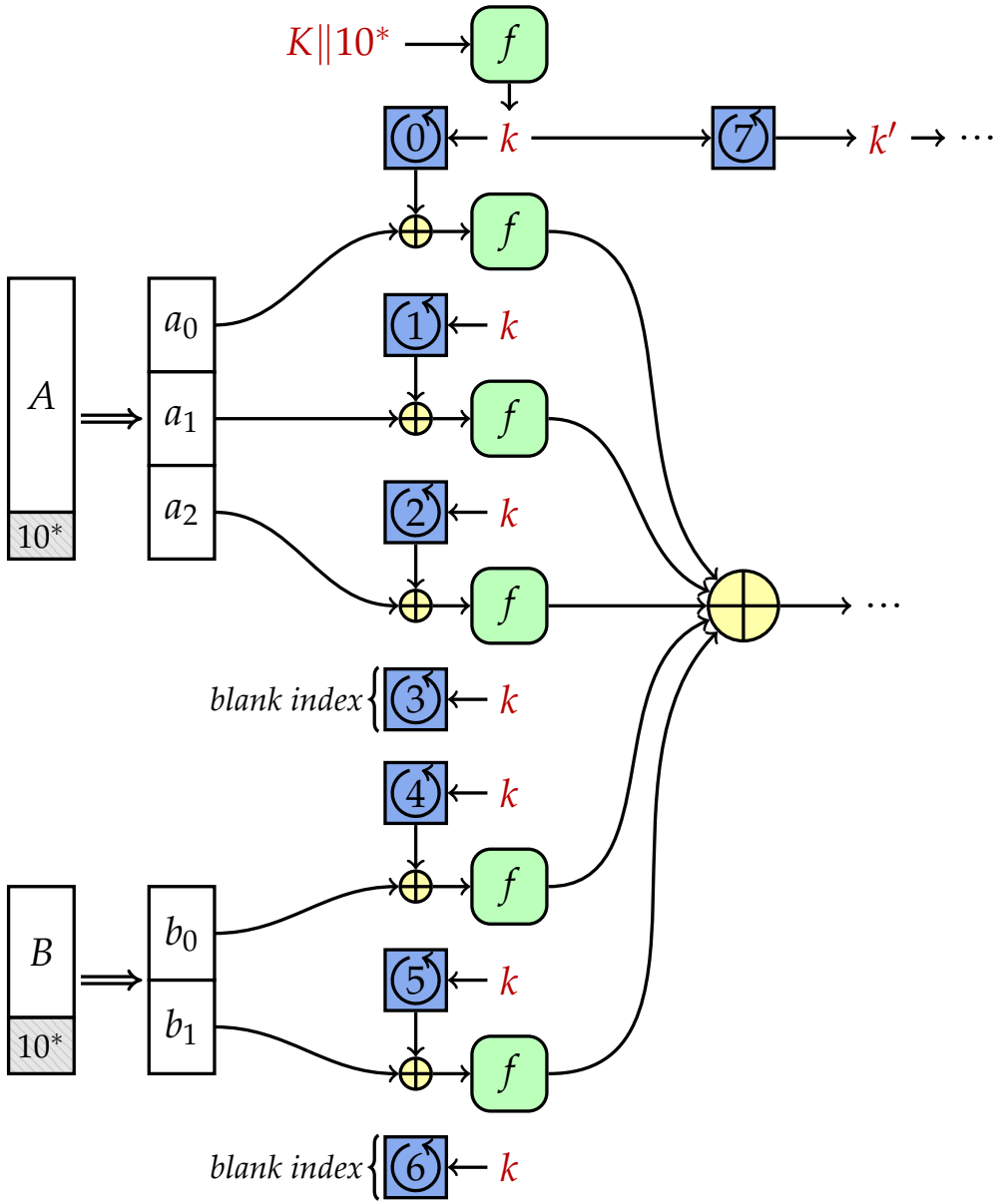
Outline

Farfalle



[FSE 2018]

Multi-string input and incrementality



Outline

Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K\left(X^{(m)} \circ \cdots \circ X^{(1)}\right) \ll q$$

doubly extendable cryptographic keyed function

Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K\left(X^{(m)} \circ \cdots \circ X^{(1)}\right) \ll q$$

- Input: sequence of strings $X^{(m)} \circ \cdots \circ X^{(1)}$

Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K\left(X^{(m)} \circ \cdots \circ X^{(1)}\right) \ll q$$

- Input: sequence of strings $X^{(m)} \circ \cdots \circ X^{(1)}$
- Output: potentially infinite output
  - **pseudo-random function of the input**
  - taking n bits starting from offset q

Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K\left(X^{(m)} \circ \cdots \circ X^{(1)}\right) \ll q$$

Efficient incrementality

- Extendable input
  1. Compute $F_K(X)$
  2. Compute $F_K(Y \circ X)$: cost independent of X

Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K\left(X^{(m)} \circ \cdots \circ X^{(1)}\right) \ll q$$

Efficient incrementality

- Extendable input
  1. Compute $F_K(X)$
  2. Compute $F_K(Y \circ X)$: cost independent of X
- Extendable output
  1. Request n_1 bits from offset 0
  2. Request n_2 bits from offset n_1: cost independent of n_1

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Deck-SANE: session-supporting and nonce-based

Initialization taking nonce $N \in \mathbb{Z}_2^*$
$e \leftarrow 0^1$
history $\leftarrow N$
return optional setup tag $T = 0^t + F_K(\text{history})$

Wrap taking metadata $A \in \mathbb{Z}_2^*$ and plaintext $P \in \mathbb{Z}_2^*$
$C \leftarrow P + F_K(\text{history}) \ll t$
history $\leftarrow A||0||e \circ$ history
history $\leftarrow C||1||e \circ$ history
$T \leftarrow 0^t + F_K(\text{history})$
$e \leftarrow e + 1^1$
return ciphertext C and tag T

Other applications

Using a deck function:

- Deck-SANE: session AE relying on user nonce
- Deck-SANSE: session AE using SIV technique
- Deck-WBC: tweakable wide block cipher

Any questions?

Thanks for your attention!

`https://keccak.team/`