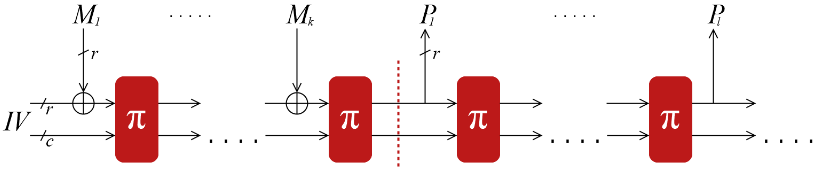
The Parazoa Family: Generalizing the Sponge Hash Functions

Elena Andreeva, Bart Mennink and Bart Preneel
KU Leuven

ECRYPT II Hash Workshop 2011 — May 19, 2011

The Sponge Hash Function Design



1. Message padded into $M_1, \ldots, M_k$ (where $M_k \neq 0$)
2. M_i's iteratively compressed in the absorbing phase
3. P_i's iteratively extracted in the extraction phase
4. $P_1, \ldots, P_l$ are concatenated and chopped if necessary

- Sponge functions indifferentiable from RO up to $O(2^{c/2})$ queries

Sponge Functions and Variants

- Sponge function:
    - Keccak

Sponge Functions and Variants

- Sponge function:
  - Keccak
- "Sponge-like" functions:
  - Grindahl
  - SHA-3 candidates CubeHash, Fugue, Hamsi, JH, Luffa

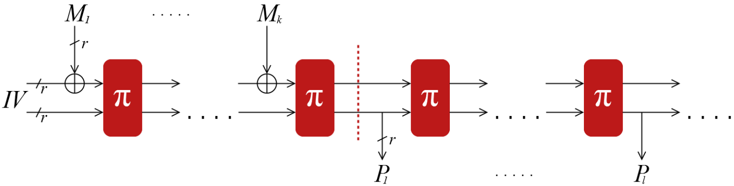
Sponge Functions and Variants

- Sponge function:
  - Keccak
- "Sponge-like" functions:
  - Grindahl
  - SHA-3 candidates CubeHash, Fugue, Hamsi, JH, Luffa

- Security of sponge functions does not directly carry over
- Minor modification to sponge design can make it insecure

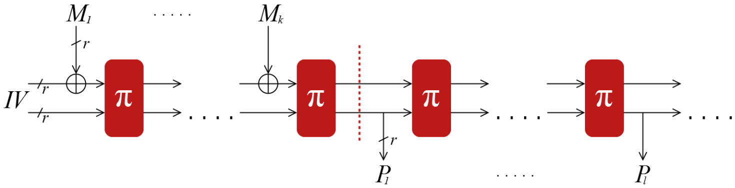
Insecure Sponge-Like Function

A sponge-like design (here, $c = r$):



Insecure Sponge-Like Function

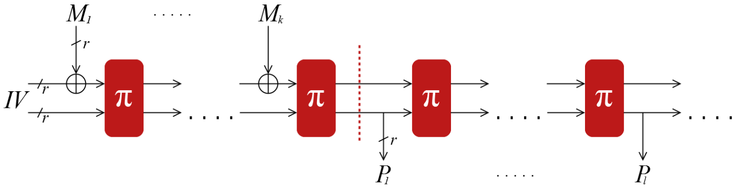
A sponge-like design (here, $c = r$):



- Differentiable from RO due to the length-extension attack
- Injection into upper halve, extraction from lower halve

Insecure Sponge-Like Function

A sponge-like design (here, $c = r$):



- Differentiable from RO due to the length-extension attack
- Injection into upper halve, extraction from lower halve

- Attack does not invalidate security of the original sponge design

Origin of the Name "Parazoa"

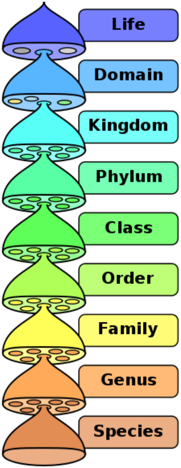
Sponge

Source: http://en.wikipedia.org/wiki/Parazoa

Origin of the Name "Parazoa"

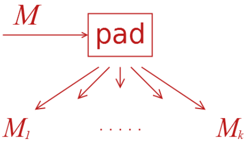
Sponge

In the biological classification of organisms, sponges are a member of the phylum Porifera, which belongs to the subkingdom Parazoa



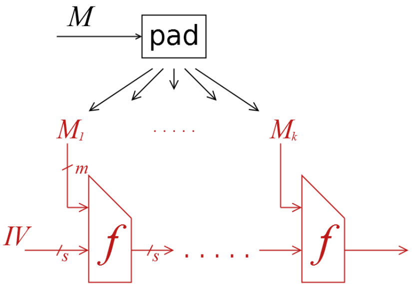
Source: http://en.wikipedia.org/wiki/Parazoa

The Parazoa Hash Function Design



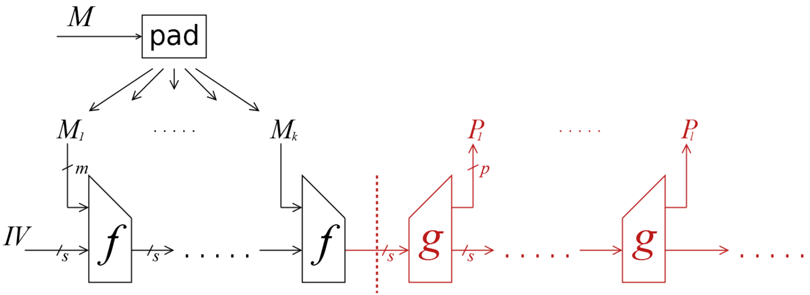
1. M padded into $M_1, \ldots, M_k$

The Parazoa Hash Function Design



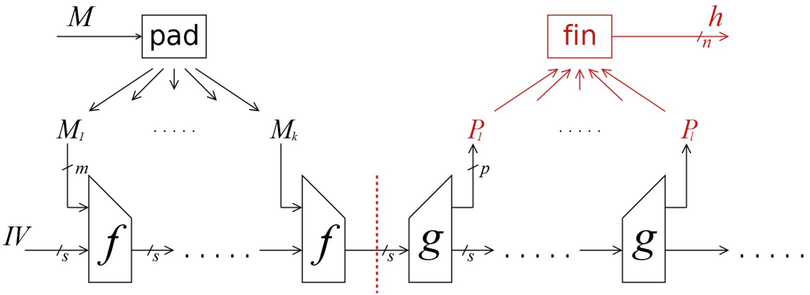
1. M padded into $M_1, \ldots, M_k$
2. M_i's iteratively compressed in the absorbing phase

The Parazoa Hash Function Design



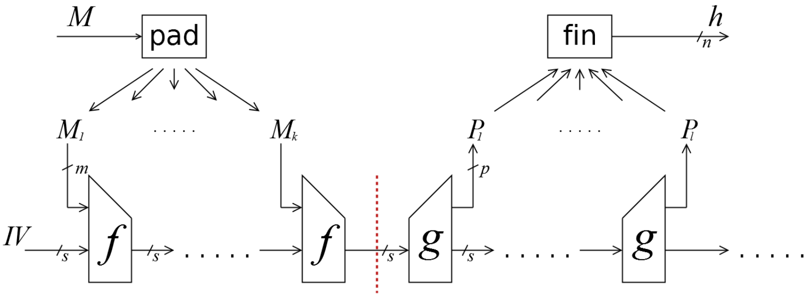
1. M padded into $M_1, \ldots, M_k$
2. M_i's iteratively compressed in the absorbing phase
3. P_i's iteratively extracted in the extraction phase

The Parazoa Hash Function Design



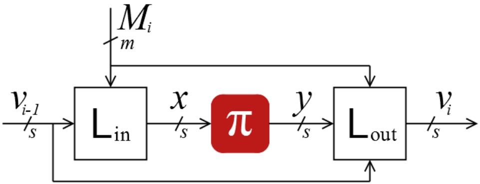
1. M padded into $M_1, \ldots, M_k$
2. M_i's iteratively compressed in the absorbing phase
3. P_i's iteratively extracted in the extraction phase
4. h generated from $P_1, \ldots, P_l$ in the finalization

The Parazoa Hash Function Design

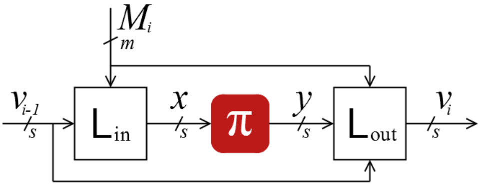


- The functions f, g, fin and pad are discussed in more detail
- π is an s-bits permutation
  - Assumed to behave like random primitive

Compression Function f



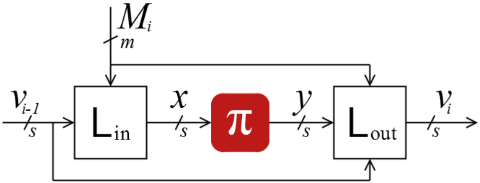
Compression Function f



We require:

- For fixed v_{i-1}, a distinct M_i results in a distinct $x = \mathsf{L}_{\text{in}}(v_{i-1}, M_i)$

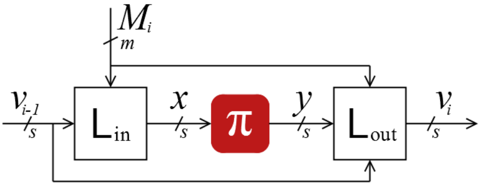
Compression Function f



We require:

- For fixed v_{i-1}, a distinct M_i results in a distinct $x = \mathsf{L}_{\text{in}}(v_{i-1}, M_i)$
- If x, x' share some preimage v_{i-1} under L_{in}, they share all preimages

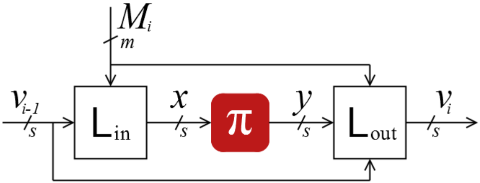
Compression Function f



We require:

- For fixed v_{i-1}, a distinct M_i results in a distinct $x = \mathsf{L}_{\text{in}}(v_{i-1}, M_i)$
- If x, x' share some preimage v_{i-1} under L_{in}, they share all preimages
- For fixed v_{i-1}, M_i, the function L_{out} is a bijection on the state

Compression Function f

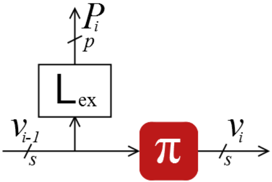


We require:

- For fixed v_{i-1}, a distinct M_i results in a distinct $x = \mathsf{L}_{\mathrm{in}}(v_{i-1}, M_i)$
- If x, x' share some preimage v_{i-1} under L_{in}, they share all preimages
- For fixed v_{i-1}, M_i, the function $\mathsf{L}_{\mathrm{out}}$ is a bijection on the state

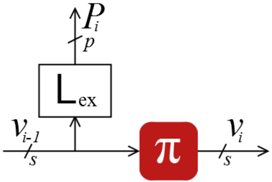
Standard functions L_{in} and $\mathsf{L}_{\mathrm{out}}$ satisfy these requirements

Extraction Function g



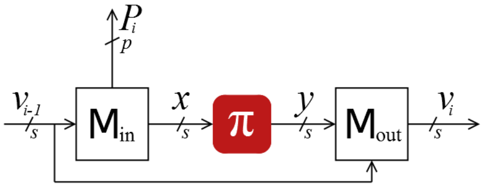
We require: L_{ex} is balanced

Extraction Function g

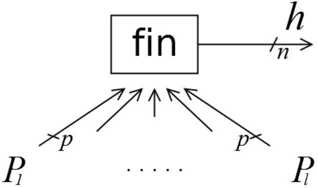


We require: L_{ex} is balanced

Result can be extended to more general g:



Finalization Function fin

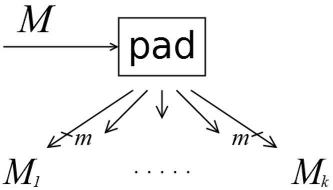


We require: fin is balanced

- Parazoa functions also allow for arbitrarily long outputs
- Sponge design:

$$\mathsf{fin}(P_1, \ldots, P_l) = \mathsf{chop}_{lp-n}(P_1 \| \cdots \| P_l)$$

Padding Function pad



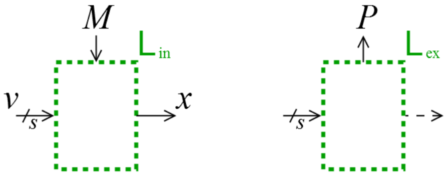
We require: pad is any injective padding function s.t.:

- Either $l = 1$ (only one extraction round), or
- Last block M_k satisfies for any x, v', M':

$$\mathsf{L}_{\text{in}}(x, M_k) \neq x \text{ and } \mathsf{L}_{\text{in}}(\mathsf{L}_{\text{out}}(x, v', M'), M_k) \neq x$$

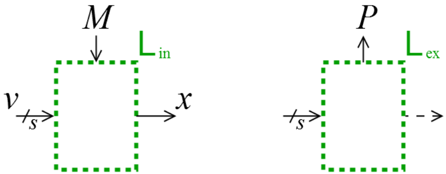
(for sponge functions: "last block is non-zero")

Parameter d



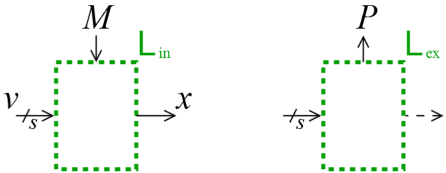
- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M

Parameter d



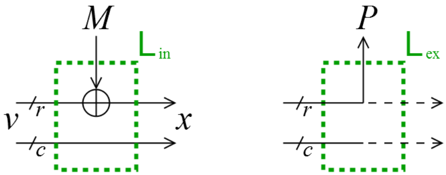
- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)

Parameter d



- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)
- Intuitively, $s - d - p$ corresponds to the "capacity"

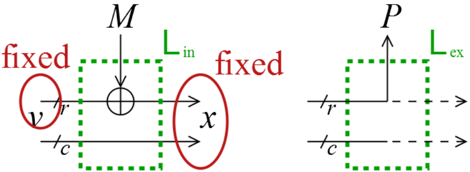
Parameter d



- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)
- Intuitively, $s - d - p$ corresponds to the "capacity"

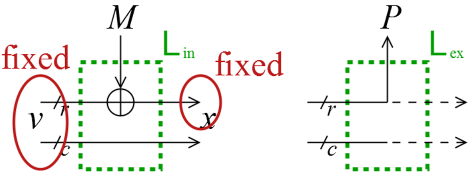
- For sponge functions:

Parameter d



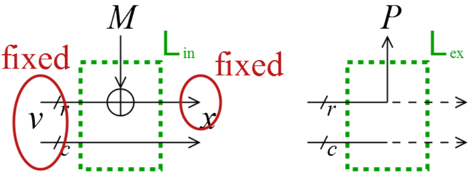
- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)
- Intuitively, $s - d - p$ corresponds to the "capacity"
- For sponge functions:

Parameter d



- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)
- Intuitively, $s - d - p$ corresponds to the "capacity"
- For sponge functions:

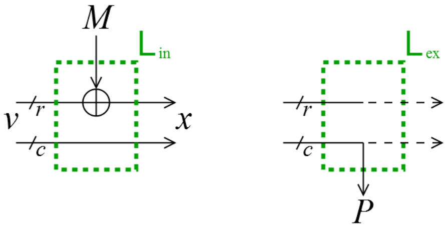
Parameter d



- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)
- Intuitively, $s - d - p$ corresponds to the "capacity"

- For sponge functions: $d = 0$ and $s - d - p = c$

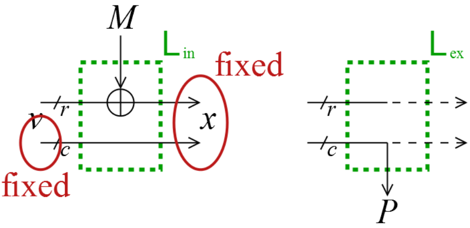
Parameter d



- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)
- Intuitively, $s - d - p$ corresponds to the "capacity"

- For sponge functions: $d = 0$ and $s - d - p = c$
- For the insecure sponge-like function:

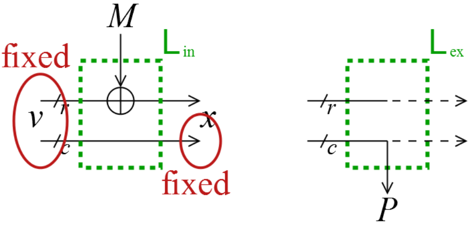
Parameter d



- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)
- Intuitively, $s - d - p$ corresponds to the "capacity"

- For sponge functions: $d = 0$ and $s - d - p = c$
- For the insecure sponge-like function:

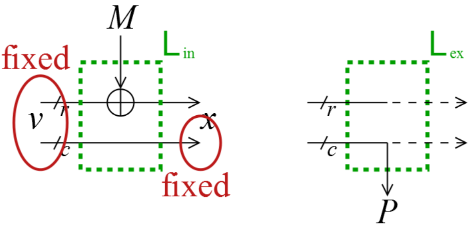
Parameter d



- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)
- Intuitively, $s - d - p$ corresponds to the "capacity"

- For sponge functions: $d = 0$ and $s - d - p = c$
- For the insecure sponge-like function:

Parameter d



- Consider tuples (v, x) s.t. $\mathsf{L}_{\text{in}}(v, M) = x$ for some M
- $d \geq 0$ is the minimal value such that:
  - For fixed x and $P := \mathsf{L}_{\text{ex}}(v)$: at most 2^d possible tuples (v, x)
  - For fixed v and $P := \mathsf{L}_{\text{ex}}(x)$: at most 2^d possible tuples (v, x)
- Intuitively, $s - d - p$ corresponds to the "capacity"

- For sponge functions: $d = 0$ and $s - d - p = c$
- For the insecure sponge-like function: $d = r$ and $s - d - p = 0$

Security Analysis

Parazoa functions are $O\left(\frac{(Kq)^2}{2^{s-d-p}}\right)$ indifferentiable from RO

(where the distinguisher makes at most q queries of K blocks)

- s: iterated state size
- d: quantity inherent to the specific parazoa design
- p: number of bits extracted in one execution of g

Security Analysis

Parazoa functions are $O\left(\frac{(Kq)^2}{2^{s-d-p}}\right)$ indifferentiable from RO

(where the distinguisher makes at most q queries of K blocks)

s: iterated state size

d: quantity inherent to the specific parazoa design

p: number of bits extracted in one execution of g

- π behaves like a random permutation
- Result can be generalized to use of multiple random primitives

Implications for Existing Designs

Algorithm	(s, m, p)	d	Indiff. $q \approx$	Assumption
Sponge	$(r+c, r, r)$	0	$2^{c/2}$	π ideal
Grindahl	(s, m, n)	m	$2^{(s-m-n)/2}$	π ideal
Quark	$(r+c, r, r)$	0	$2^{c/2}$	π ideal
PHOTON-$(r' \leq r)$	$(r+c, r, r')$	$r-r'$	$2^{c/2}$	π ideal
PHOTON-$(r' \geq r)$	$(r+c, r, r')$	0	$2^{(c+r-r')/2}$	π ideal
SPONGENT	$(r+c, r, r)$	0	$2^{c/2}$	π ideal
CubeHash-n	$(1024, 257, n)$	1	$2^{(1023-n)/2}$	P^{16} ideal
Fugue-$(n \leq 256)$	$(960, 32, n)$	m	$2^{(928-n)/2}$	π, π' ideal
Fugue-$(n > 256)$	$(1152, 32, n)$	m	$2^{(1120-n)/2}$	π, π' ideal
JH-n	$(1024, 512, n)$	m	$2^{(512-n)/2}$	π ideal
Keccak-n	$(1600, s-2n, n)$	$s-3n$	2^n	π ideal
Luffa-$(n \leq 256)$	$(768, 256, 256)$	0	2^{256}	$Q_1 \| \cdots \| Q_3$ ideal
Luffa-384	$(1024, 256, 256)$	0	2^{384}	$Q_1 \| \cdots \| Q_4$ ideal
Luffa-512	$(1280, 256, 256)$	0	2^{512}	$Q_1 \| \cdots \| Q_5$ ideal

s = internal state, m = message injection, p = is digest extraction, n = output size

For SHA-3 candidates: $n \in \{224, 256, 384, 512\}$

Implications for Existing Designs

Algorithm	(s,m,p)	d	Indiff. $q \approx$	Assumption
Sponge	$(r+c,r,r)$	0	$2^{c/2}$	π ideal
Grindahl	(s,m,n)	m	$2^{(s-m-n)/2}$	π ideal
Quark	$(r+c,r,r)$	0	$2^{c/2}$	π ideal
PHOTON-($r' \leq r$)	$(r+c,r,r')$	$r-r'$	$2^{c/2}$	π ideal
PHOTON-($r' \geq r$)	$(r+c,r,r')$	0	$2^{(c+r-r')/2}$	π ideal
SPONGENT	$(r+c,r,r)$	0	$2^{c/2}$	π ideal
CubeHash-n	$(1024,257,n)$	1	$2^{(1023-n)/2}$	P^{16} ideal
Fugue-($n \leq 256$)	$(960,32,n)$	m	$2^{(928-n)/2}$	π,π' ideal
Fugue-($n > 256$)	$(1152,32,n)$	m	$2^{(1120-n)/2}$	π,π' ideal
JH-n	$(1024,512,n)$	m	$2^{(512-n)/2}$	π ideal
Keccak-n	$(1600,s-2n,n)$	$s-3n$	2^n	π ideal
Luffa-($n \leq 256$)	$(768,256,256)$	0	2^{256}	$Q_1\|\cdots\|Q_3$ ideal
Luffa-384	$(1024,256,256)$	0	2^{384}	$Q_1\|\cdots\|Q_4$ ideal
Luffa-512	$(1280,256,256)$	0	2^{512}	$Q_1\|\cdots\|Q_5$ ideal

s = internal state, m = message injection, p = is digest extraction, n = output size
For SHA-3 candidates: $n \in \{224,256,384,512\}$

- Moody et al. (2012): indifferentiability of JH up to 2^{256} queries

Implications for Existing Designs

Algorithm	(s, m, p)	d	Indiff. $q \approx$	Assumption
Sponge	$(r+c, r, r)$	0	$2^{c/2}$	π ideal
Grindahl	(s, m, n)	m	$2^{(s-m-n)/2}$	π ideal
Quark	$(r+c, r, r)$	0	$2^{c/2}$	π ideal
PHOTON-$(r' \leq r)$	$(r+c, r, r')$	$r-r'$	$2^{c/2}$	π ideal
PHOTON-$(r' \geq r)$	$(r+c, r, r')$	0	$2^{(c+r-r')/2}$	π ideal
SPONGENT	$(r+c, r, r)$	0	$2^{c/2}$	π ideal
CubeHash-n	$(1024, 257, n)$	1	$2^{(1023-n)/2}$	P^{16} ideal
Fugue-$(n \leq 256)$	$(960, 32, n)$	m	$2^{(928-n)/2}$	π, π' ideal
Fugue-$(n > 256)$	$(1152, 32, n)$	m	$2^{(1120-n)/2}$	π, π' ideal
JH-n	$(1024, 512, n)$	m	$2^{(512-n)/2}$	π ideal
Keccak-n	$(1600, s-2n, n)$	$s-3n$	2^{n}	π ideal
Luffa-$(n \leq 256)$	$(768, 256, 256)$	0	2^{256}	$Q_1 \| \cdots \| Q_3$ ideal
Luffa-384	$(1024, 256, 256)$	0	2^{384}	$Q_1 \| \cdots \| Q_4$ ideal
Luffa-512	$(1280, 256, 256)$	0	2^{512}	$Q_1 \| \cdots \| Q_5$ ideal

s = internal state, m = message injection, p = is digest extraction, n = output size
For SHA-3 candidates: $n \in \{224, 256, 384, 512\}$

- Moody et al. (2012): indifferentiability of JH up to 2^{256} queries
  - Design-specific proofs may result in better bounds

Conclusions

- Parazoa hash functions: a generalization of the sponge hash functions
- Parazoa functions cover a.o. sponges, Grindahl, PHOTON, and several SHA-3 candidates

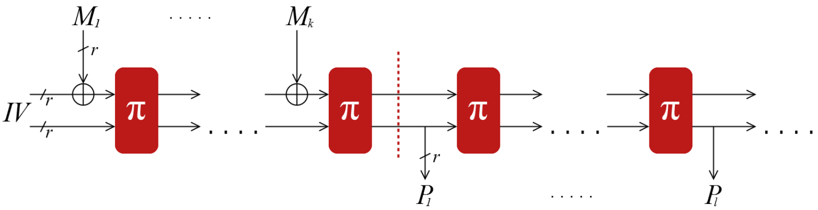
- Parazoa functions are proven indifferentiable from RO

- Further research
  - Tightness of the indifferentiability bound?
  - Improved collision/preimage resistance of the parazoa design?
  - Generalization to *animalia functions* or *eukaryota functions*?

Thank you for your attention!

Insecure sponge-like design

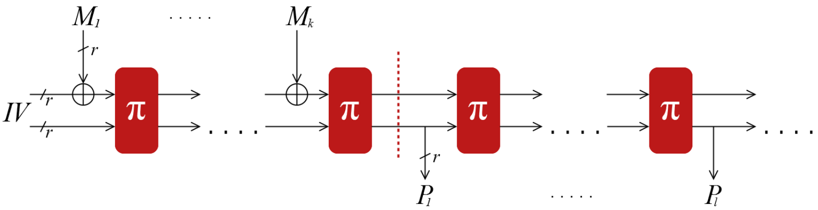
What about the insecure sponge-like design?



- This insecure sponge-like design falls within the parazoa framework

Insecure sponge-like design

What about the insecure sponge-like design?



- This insecure sponge-like design falls within the parazoa framework
- But parameter $d = s - p$, and thus $s - d - p = 0$
  $\rightarrow$ Our indifferentiability result implies $O(1)$ indifferentiability bound