Generalized Indifferentiable Sponge and its Application to Polygon Miden VM

☃️ ALPSY 2025, Obergurgl ⛷️

Tomer Ashur	**Amit Singh Bhati**
3MI Labs	3MI Labs & COSIC, KU Leuven

KU LEUVEN 3MI LABS

Random Oracle

What is a Random Oracle

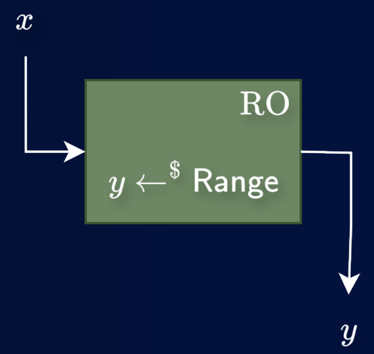
- Monolithic theoretical construct
- Infinite domain but finite range
- Uniform and consistent
- Commonly modeled as a black-box lookup table

What is a Random Oracle

- Monolithic theoretical construct
- Infinite domain but finite range
- Uniform and consistent
- Commonly modeled as a black-box lookup table

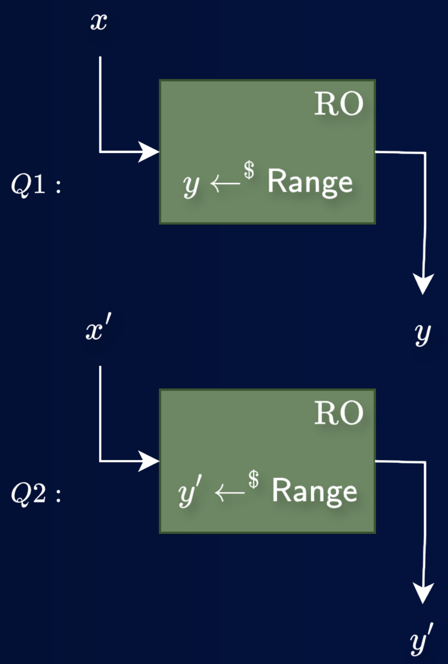
What is a Random Oracle

- Monolithic theoretical construct
- Infinite domain but finite range
- Uniform and consistent
- Commonly modeled as a black-box lookup table



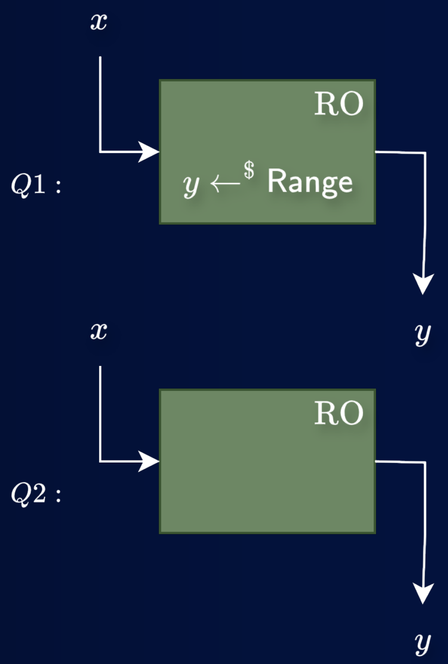
What is a Random Oracle

- Monolithic theoretical construct
- Infinite domain but finite range
- Uniform and consistent
- Commonly modeled as a black-box lookup table



What is a Random Oracle

- Monolithic theoretical construct
- Infinite domain but finite range
- Uniform and consistent
- Commonly modeled as a black-box lookup table



Applications of Random Oracle in PETs

- Digital Signatures & Threshold Cryptography

  (e.g., RSA-PSS, ECDSA, BLS, threshold BLS)

- Zero-Knowledge Proofs & Commitment Schemes

  (e.g., Fiat-Shamir Transform, Schnorr Commitment)

- Secure Multi-Party Computation (MPC) & Oblivious Transfer (OT) Extensions

  (e.g., SPDZ Protocol, IKNP and KOS OTs)

- Randomness Generation

Random Oracle: Popular Domains

$$\mathrm{RO} : \mathsf{Domain} \rightarrow \mathsf{Range}$$

Random Oracle: Popular Domains

$$\mathrm{RO} : {\mathbb{F}_Q}^* \to {\mathbb{F}_Q}^h$$

Random Oracle: Popular Domains

$$\mathrm{RO} : {\mathbb{F}_Q}^* \rightarrow {\mathbb{F}_Q}^h$$

Binary setting:

when $Q = 2^k$ for $k \geq 1$

$$\mathrm{RO}_1 : {\mathbb{F}_{2^k}}^* \rightarrow {\mathbb{F}_{2^k}}^g$$

Non-binary setting:

when $Q = p^k$ for $p \neq 2, k \geq 1$

$$\mathrm{RO}_2 : {\mathbb{F}_{p^k}}^* \rightarrow {\mathbb{F}_{p^k}}^h$$

Random Oracle: Popular Domains

$$\mathrm{RO} : \mathbb{F}_Q{}^* \to \mathbb{F}_Q{}^h$$

Binary setting:

when $Q = 2^k$ for $k \geq 1$

Non-binary setting:

when $Q = p^k$ for $p \neq 2, k \geq 1$

$$\mathrm{RO}_1 : \{0,1\}^* \to \{0,1\}^{g'}$$

$$\mathrm{RO}_2 : \mathbb{F}_p{}^* \to \mathbb{F}_p{}^{h'}$$

Real-world Instantiations?

- True random oracles do not exist in real world

$$\mathrm{RO} \xleftarrow{\$} \mathbf{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$$

- Infinite choices ➔ Infinite description
- Impossible to implement in a finite algorithm

Real-world Instantiations?

- True random oracles do not exist in real world

$$\mathrm{RO} \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$$

- Infinite choices ➔ Infinite description
- Impossible to implement in a finite algorithm

- Cryptographic hash functions (CHFs) are used to emulate

$$H \leftarrow \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$$

Real-world Instantiations?

- True random oracles do not exist in real world

$$\mathrm{RO} \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$$

- Infinite choices ➔ Infinite description
- Impossible to implement in a finite algorithm

- Cryptographic hash functions (CHFs) are used to emulate

$$H \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$$

Real-world Instantiations?

- True random oracles do not exist in real world

$$\mathrm{RO} \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$$

- Infinite choices ➔ Infinite description
- Impossible to implement in a finite algorithm

- Cryptographic hash functions (CHFs) are used to emulate

$$H \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$$

$$\text{but with a finite algorithm } \mathfrak{H}$$

Real-world Instantiations?

- True random oracles do not exist in real world

$$\mathrm{RO} \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$$

- Infinite choices ➔ Infinite description
- Impossible to implement in a finite algorithm

- Cryptographic hash functions (CHFs) are used to emulate

$$H \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$$

$$\text{but with a finite algorithm } \mathfrak{H}$$

- Efficient to implement and evaluate

Real-world Instantiations?

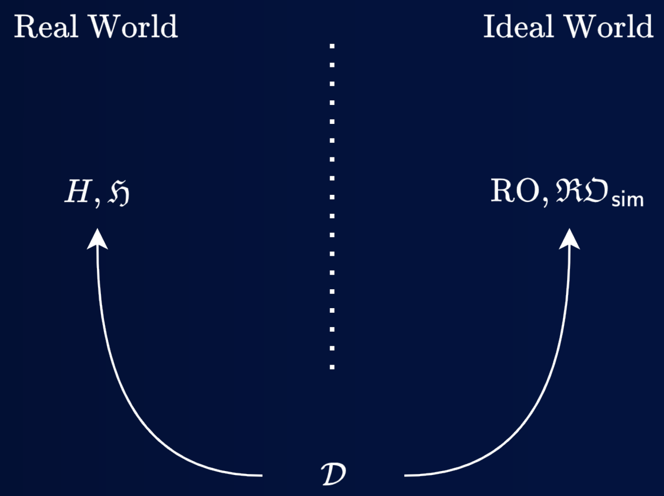
- H is *indistinguishable* from a random oracle if
  - $\mathfrak{H}$ leaks nothing *significant* about its map
  - only way to infer something *significant* about an output is evaluation

Real-world Instantiations?

- H is *indistinguishable* from a random oracle if
  - – $\mathfrak{H}$ leaks nothing *significant* about its map
  - – only way to infer something *significant* about an output is evaluation

In other words,

$\exists\, \mathfrak{RO}_{\mathsf{sim}}$ such that for any $\mathcal{D}$:



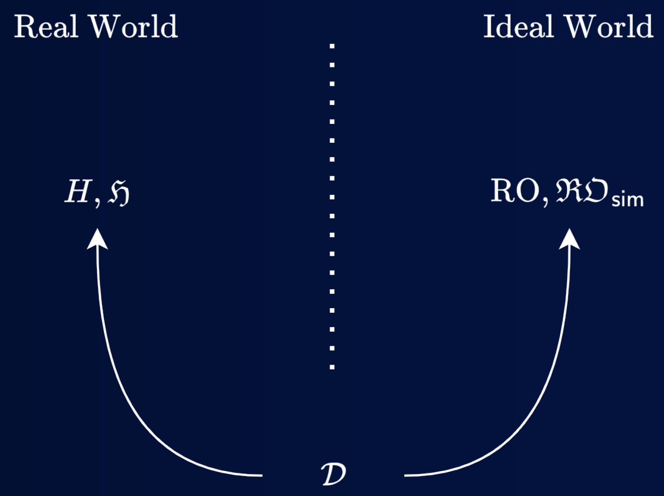
Real-world Instantiations?

- H is *indistinguishable* from a random oracle if
  - – $\mathfrak{H}$ leaks nothing *significant* about its map
  - – only way to infer something *significant* about an output is evaluation

In other words,

$\exists\, \mathfrak{RO}_{\mathsf{sim}}$ such that for any $\mathcal{D}$:

$$\mathcal{D}_q((H, \mathfrak{H}), (\mathrm{RO}, \mathfrak{RO}_{\mathsf{sim}})) \leq \epsilon_q$$



Real-world Instantiations?

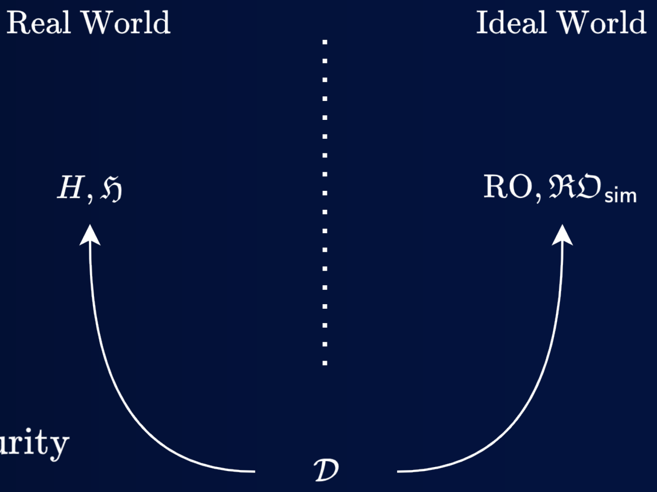
- H is *indistinguishable* from a random oracle if
  - $\mathfrak{H}$ leaks nothing *significant* about its map
  - only way to infer something *significant* about an output is evaluation

In other words,

$\exists\, \mathfrak{RO}_{\mathsf{sim}}$ such that for any $\mathcal{D}$:

$$\mathcal{D}_q((H, \mathfrak{H}), (\mathrm{RO}, \mathfrak{RO}_{\mathsf{sim}})) \leq \epsilon_q$$

reasonably *small* as per target security



Real-world Instantiations?

- impossible to simulate a finite algorithm for RO

Real-world Instantiations?

- impossible to simulate a finite algorithm for RO
- So, we can't provably achieve this notion!

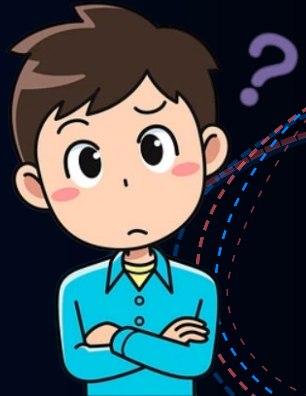
Real-world Instantiations?

- impossible to simulate a finite algorithm for RO
- So, we can't provably achieve this notion!
- Unless we make some assumptions for H

Real-world Instantiations?

- impossible to simulate a finite algorithm for RO
- So, we can't provably achieve this notion!
- Unless we make some assumptions for H
  - – E.g., monolithic/ideal primitive-based hashes

RO Indifferentiability



Random Oracle Indifferentiability [Maurer et al., TCC'04]

- Let $H^{\mathcal{P}} : \mathbb{F}_p^* \to \mathbb{F}_p^r$ be a hash based on an ideal permutation $\mathcal{P} \xleftarrow{\$} \mathrm{Perm}(\mathbb{F}_p^b, \mathbb{F}_p^b)$
- Excluding the subroutine $\mathcal{P}$ calls, it has a public finite algorithm $\mathfrak{H}$

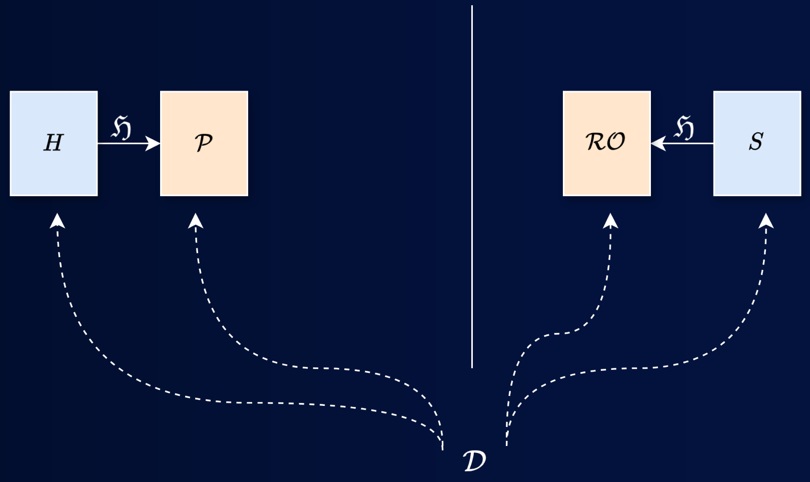
Random Oracle Indifferentiability [Maurer et al., TCC'04]

- Let $H^{\mathcal{P}} : \mathbb{F}_p^* \to \mathbb{F}_p^r$ be a hash based on an ideal permutation $\mathcal{P} \xleftarrow{\$} \mathrm{Perm}(\mathbb{F}_p^b, \mathbb{F}_p^b)$
- Excluding the subroutine $\mathcal{P}$ calls, it has a public finite algorithm $\mathfrak{H}$
- H is indifferentiable from a random oracle i.e., $\mathcal{RO} \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$ if

Random Oracle Indifferentiability [Maurer et al., TCC'04]

- Let $H^{\mathcal{P}} : \mathbb{F}_p^* \to \mathbb{F}_p^r$ be a hash based on an ideal permutation $\mathcal{P} \xleftarrow{\$} \mathrm{Perm}(\mathbb{F}_p^b, \mathbb{F}_p^b)$
- Excluding the subroutine $\mathcal{P}$ calls, it has a public finite algorithm $\mathfrak{H}$
- H is indifferentiable from a random oracle i.e., $\mathcal{RO} \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$ if

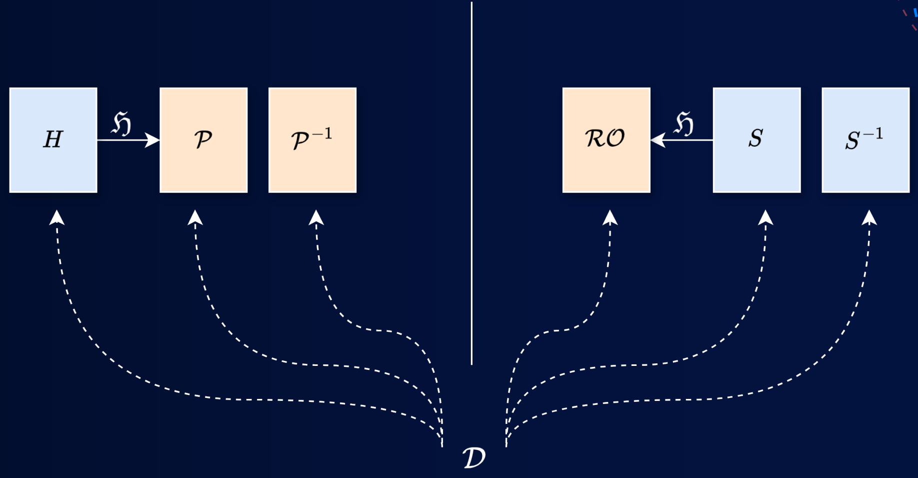
$\exists\, S$ such that for any $\mathcal{D}$:



Random Oracle Indifferentiability [Maurer et al., TCC'04]

- Let $H^{\mathcal{P}} : \mathbb{F}_p^* \to \mathbb{F}_p^r$ be a hash based on an ideal permutation $\mathcal{P} \xleftarrow{\$} \mathrm{Perm}(\mathbb{F}_p^b, \mathbb{F}_p^b)$
- Excluding the subroutine $\mathcal{P}$ calls, it has a public finite algorithm $\mathfrak{H}$
- H is indifferentiable from a random oracle i.e., $\mathcal{RO} \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$ if

$\exists\, S$ such that for any $\mathcal{D}$:



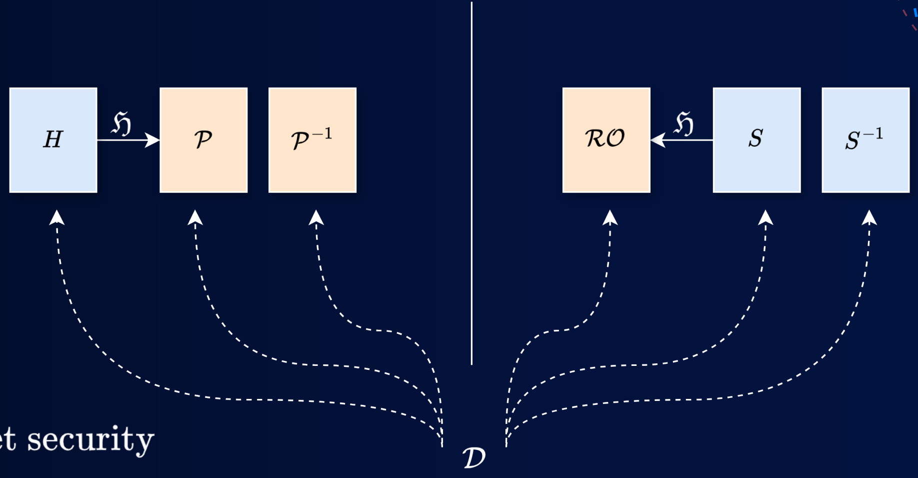
Random Oracle Indifferentiability [Maurer et al., TCC'04]

- Let $H^{\mathcal{P}} : \mathbb{F}_p^* \to \mathbb{F}_p^r$ be a hash based on an ideal permutation $\mathcal{P} \xleftarrow{\$} \mathrm{Perm}(\mathbb{F}_p^b, \mathbb{F}_p^b)$
- Excluding the subroutine $\mathcal{P}$ calls, it has a public finite algorithm $\mathfrak{H}$
- H is indifferentiable from a random oracle i.e., $\mathcal{RO} \xleftarrow{\$} \mathrm{Func}(\mathbb{F}_p^*, \mathbb{F}_p^r)$ if

$\exists\, S$ such that for any $\mathcal{D}$:

$$\mathcal{D}_q((H, \mathcal{P}, \mathcal{P}^{-1}), (\mathcal{RO}, S, S^{-1})) \le \epsilon_q$$

reasonably *small* as per target security



Implications

- Indifferentiability reduces security of H as random oracle to P as ideal permutation

Implications

- Indifferentiability reduces security of H as random oracle to P as ideal permutation

- Indifferentiability + large input-output spaces ➔ basic CHF’s properties

  e.g., collision resistance and (second)-pre-image resistance

Sponge and its Generalization

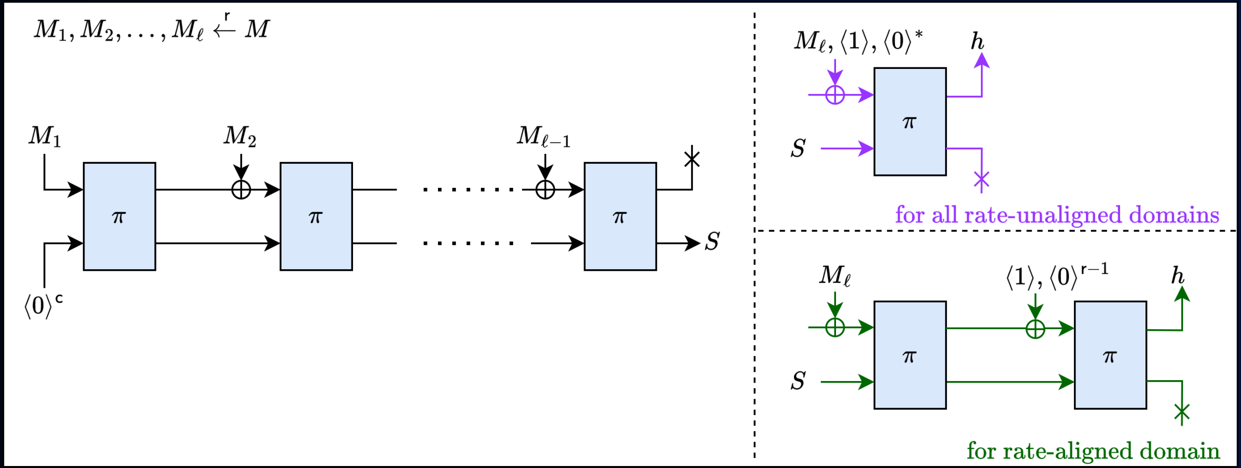
Sponge Hash Function [Bertoni et al., Ecrypt'07]

- $\text{Sponge}^{\pi} : \mathbb{F}_p^* \to \mathbb{F}_p^{\mathsf{r}}$ is the mode behind SHA-3 family
- Operates on a public permutation $\pi : \mathbb{F}_p^{\mathsf{r}+\mathsf{c}} \to \mathbb{F}_p^{\mathsf{r}+\mathsf{c}}$
- Currently deployed in many privacy-preserving applications
- Proven RO-Indifferentiable for $p = 2$ under ideal permutation model

Sponge Hash Function [Bertoni et al., Ecrypt'07]

- $\mathsf{Sponge}^{\pi} : \mathbb{F}_p^* \to \mathbb{F}_p^{\mathsf{r}}$ is the mode behind SHA-3 family

- Operates on a public permutation $\pi : \mathbb{F}_p^{\mathsf{r}+\mathsf{c}} \to \mathbb{F}_p^{\mathsf{r}+\mathsf{c}}$

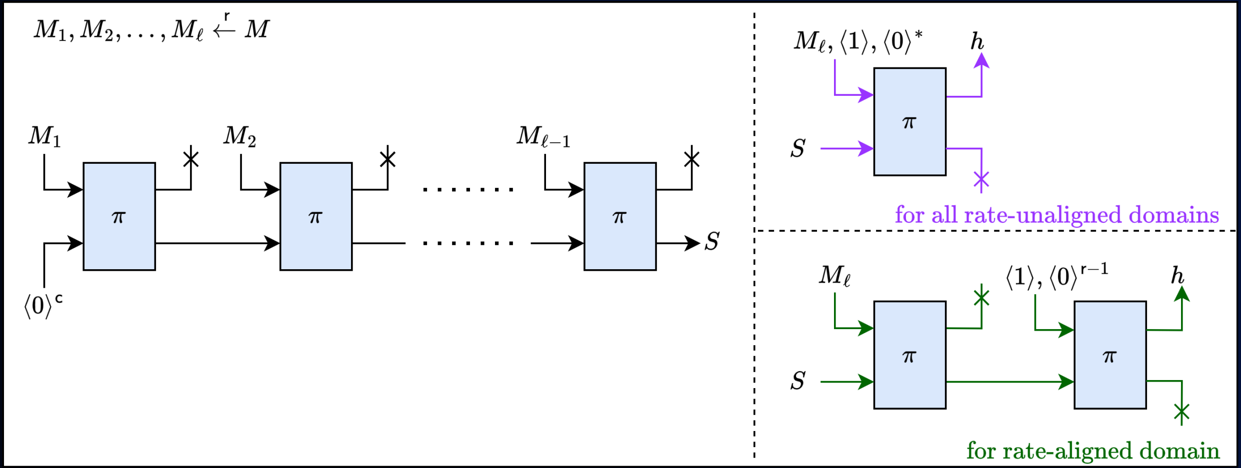


(a) Sponge ($p = 2$)

Sponge Hash Function [Bertoni et al., Ecrypt'07]

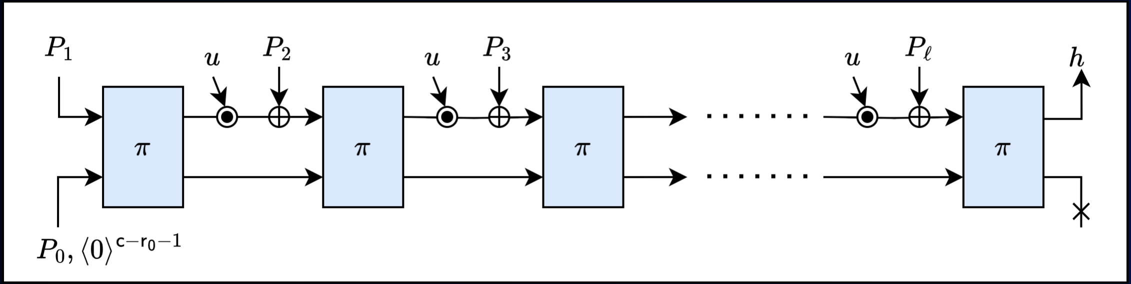
- $\mathsf{Sponge}^\pi : \mathbb{F}_p^* \to \mathbb{F}_p^{\mathsf{r}}$ is the mode behind SHA-3 family

- Operates on a public permutation $\pi : \mathbb{F}_p^{\mathsf{r}+\mathsf{c}} \to \mathbb{F}_p^{\mathsf{r}+\mathsf{c}}$



(b) Overwrite Sponge ($p = 2$)

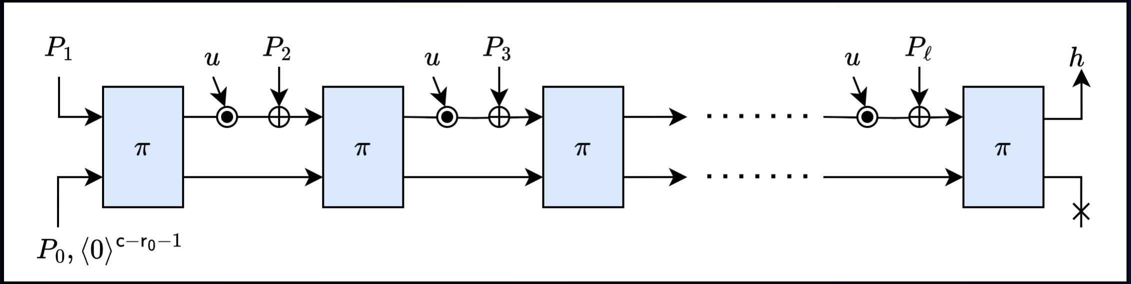
Generalizing Sponge



(c) $\mathsf{GSponge}[u, \mathsf{r}_0, p, \mathsf{pad}]$ Hashing Mode

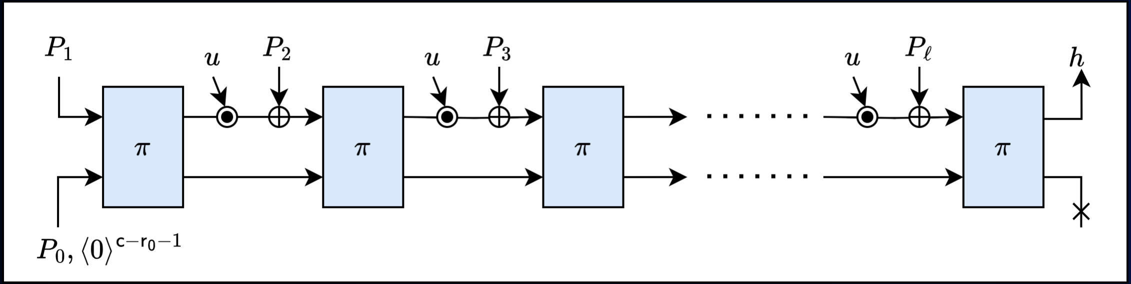
- For 1) input chaining style, 2) field setting, and 3) injective pre-/post- padding types
- $P = \mathsf{pad}(M) = \langle x, M, y \rangle, \quad u \in \{0, 1\}$

Generalizing Sponge



(c) $\mathsf{GSponge}[u, \mathsf{r}_0, p, \mathsf{pad}]$ Hashing Mode

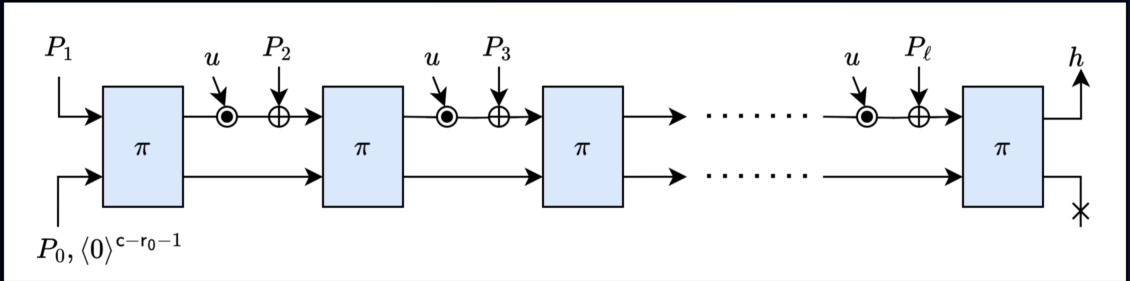
Generalizing Sponge



(c) $\mathsf{GSponge}[u, \mathsf{r}_0, p, \mathsf{pad}]$ Hashing Mode

- $\mathsf{GSponge}[1, 0, 2, \langle 0, M, 1, 0^* \rangle] =$ Sponge
- $\mathsf{GSponge}[0, 0, 2, \langle 0, M, 1, 0^* \rangle] =$ Overwrite Sponge

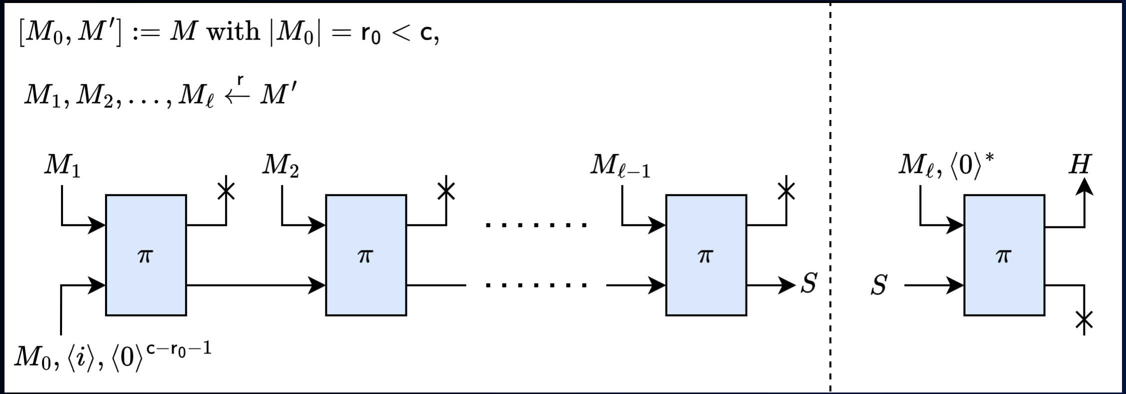
Generalizing Sponge



(c) $\mathsf{GSponge}[u, \mathsf{r}_0, p, \mathsf{pad}]$ Hashing Mode

- $\mathsf{GSponge}[1, 0, 2, \langle 0, M, 1, 0^* \rangle] = \text{Sponge}$
- $\mathsf{GSponge}[0, 0, 2, \langle 0, M, 1, 0^* \rangle] = \text{Overwrite Sponge}$
- $\mathsf{GSponge}[0, \mathsf{r}_0, p, \langle i, M, 0^i \rangle] = \mathsf{Sponge2}$

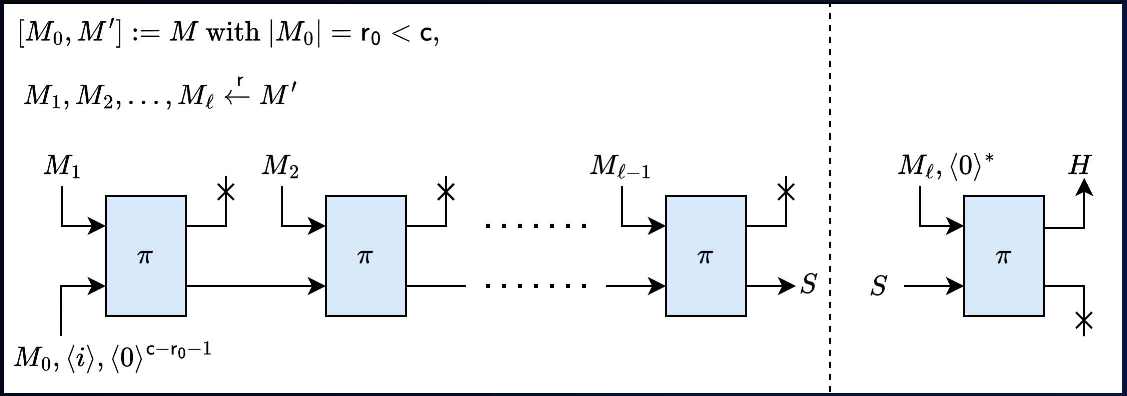
Sponge2: An Efficient GSponge Instantiation



(d) Sponge2 Hashing Mode

- $\mathsf{GSponge}[0, 0, 2, \langle 0, M, 1, 0^* \rangle] =$ Overwrite Sponge
- $\mathsf{GSponge}[0, \mathsf{r}_0, p, \langle i, M, 0^i \rangle] =$ Sponge2

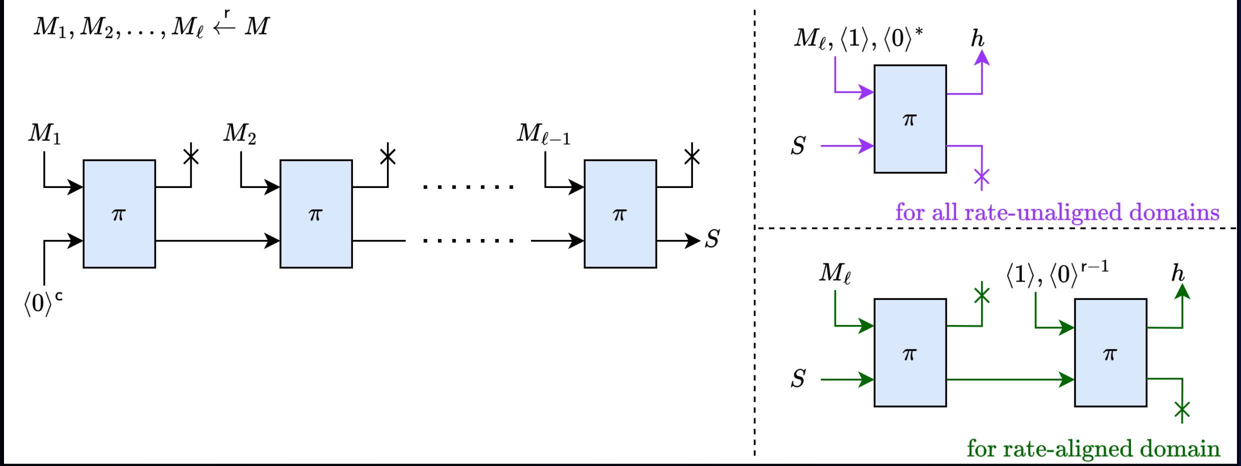
Sponge2: An Efficient GSponge Instantiation



(d) Sponge2 Hashing Mode

- $\text{GSponge}[0, 0, 2, \langle 0, M, 1, 0^* \rangle] =$ Overwrite Sponge
- $\text{GSponge}[0, r_0, p, \langle i, M, 0^i \rangle] =$ Sponge2

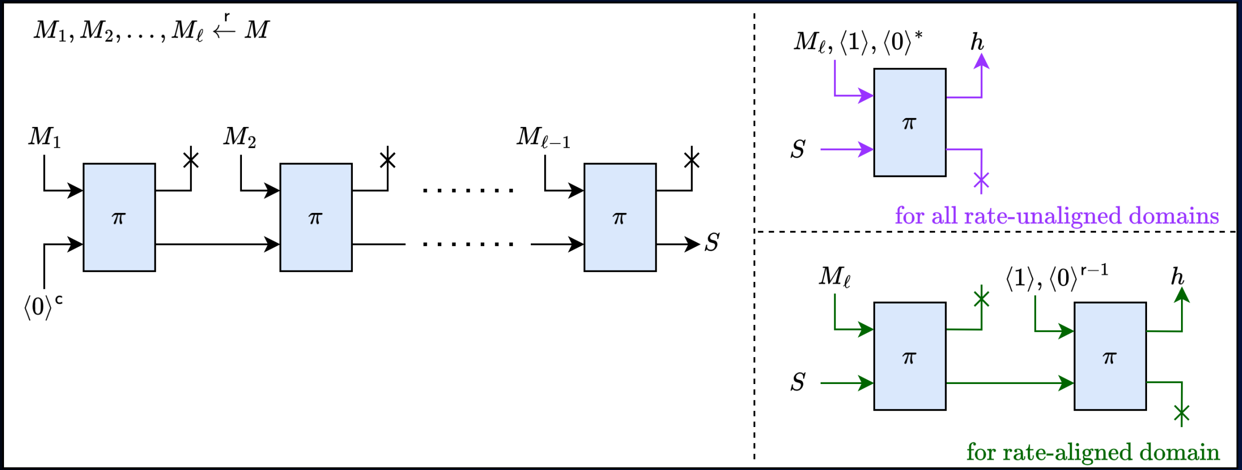
Sponge2: An Efficient GSponge Instantiation



(b) Overwrite Sponge

- GSponge$[0, 0, 2, \langle 0, M, 1, 0^* \rangle]$ = Overwrite Sponge
- GSponge$[0, r_0, p, \langle i, M, 0^i \rangle]$ = Sponge2

Sponge2: An Efficient GSponge Instantiation



(b) Overwrite Sponge

- GSponge$[0, 0, 2, \langle 0, M, 1, 0^* \rangle]$ = Overwrite Sponge
- GSponge$[0, r_0, p, \langle i, M, 0^i \rangle]$ = Sponge2

Applications of Sponge2 in Miden VM

Miden VM used RPO (under Sponge with $r = 8$) for hashing

Now shifted to RPO (under Sponge2 with $r = 8$, $r_0 = 2$) to get

Applications of Sponge2 in Miden VM

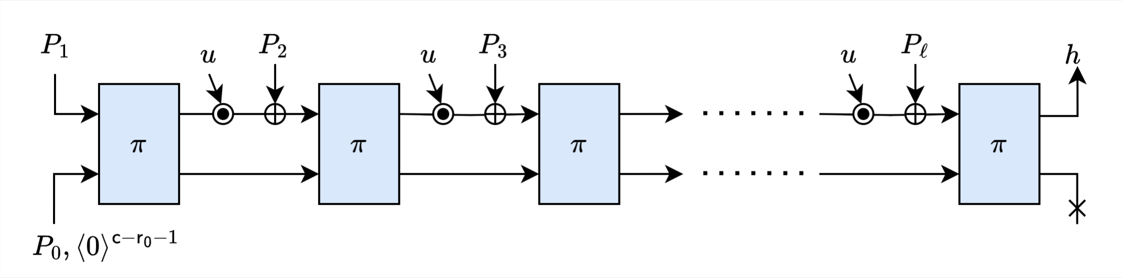
Miden VM used RPO (under Sponge with r = 8) for hashing

Now shifted to RPO (under Sponge2 with $r = 8$, $r_0 = 2$) to get

- 50% improved rate over Sponge in *2-to-1 Hashing with Metadata*
- 12.5% improved rate over Sponge in *Hashing for Leaf Computation*
- Support of Multi-rate and Multi-protocol applications of Sponge2

GSponge: Indifferentiability

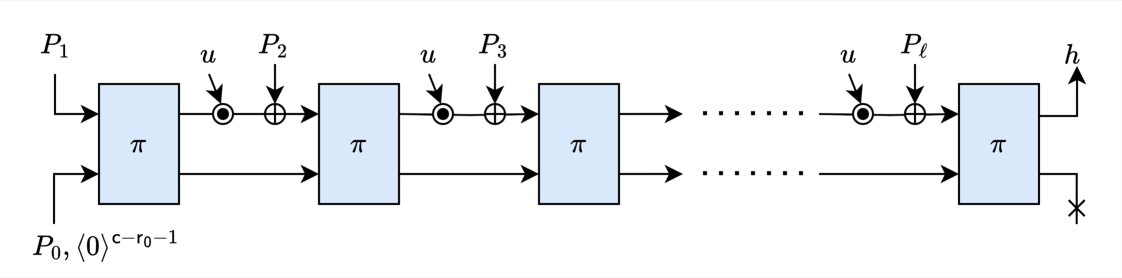
RO-Indifferentiability of GSponge



$\mathsf{GSponge}[0, r_0, p, \mathsf{pad}]$

$\longrightarrow \mathsf{GSponge}[1, r_0, p, \mathsf{pad}]$

RO-Indifferentiability of GSponge



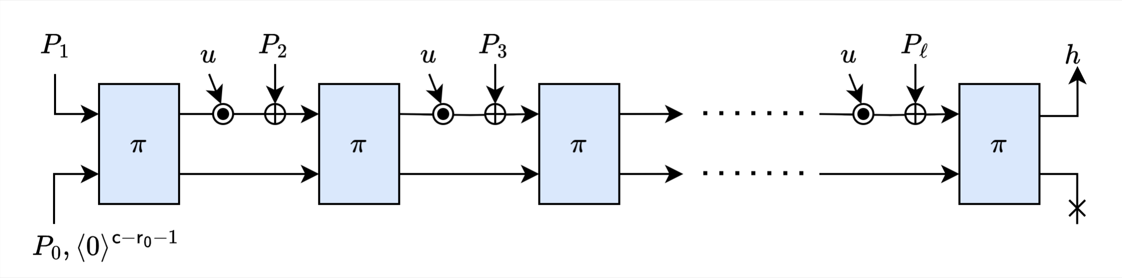
$\mathsf{GSponge}[0, r_0, p, \mathsf{pad}]$

$\longrightarrow$ $\mathsf{GSponge}[1, r_0, p, \mathsf{pad}]$

permuted but same output multiset

$\Rightarrow$ same output distribution

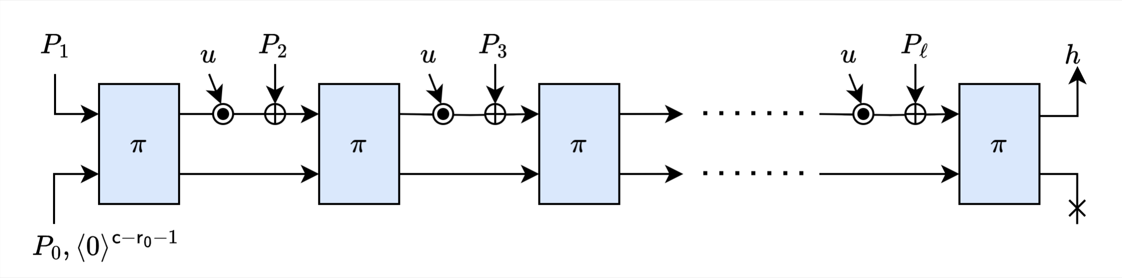
RO-Indifferentiability of GSponge



GSponge$[0, r_0, p, \mathsf{pad}]$ is RO-Indiff

→ GSponge$[1, r_0, p, \mathsf{pad}]$ is RO-Indiff

RO-Indifferentiability of GSponge

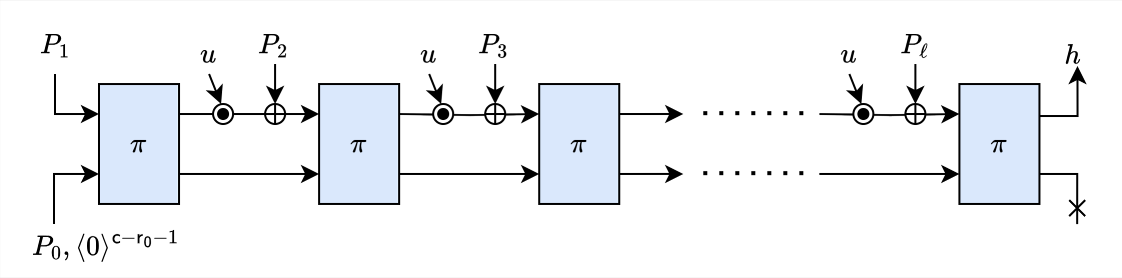


$\mathsf{GSponge}[0, r_0, p, \mathsf{pad}]$ is RO-Indiff

→ $\mathsf{GSponge}[1, r_0, p, \mathsf{pad}]$ is RO-Indiff

→ $\mathsf{GSponge}[1, r_0, p, \mathsf{pad}']$ is RO-Indiff

RO-Indifferentiability of GSponge

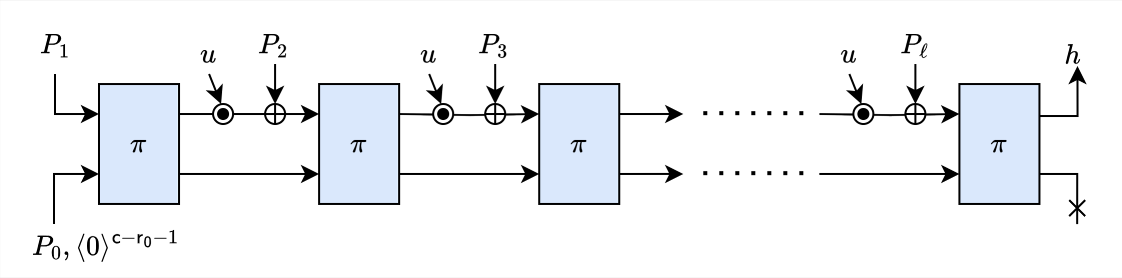


$\mathsf{GSponge}[0, r_0, p, \mathsf{pad1}]$ is RO-Indiff

(for some injective $\mathsf{pad1}$)

→ $\mathsf{GSponge}[u, r_0, p, \mathsf{pad}]$ is RO-Indiff

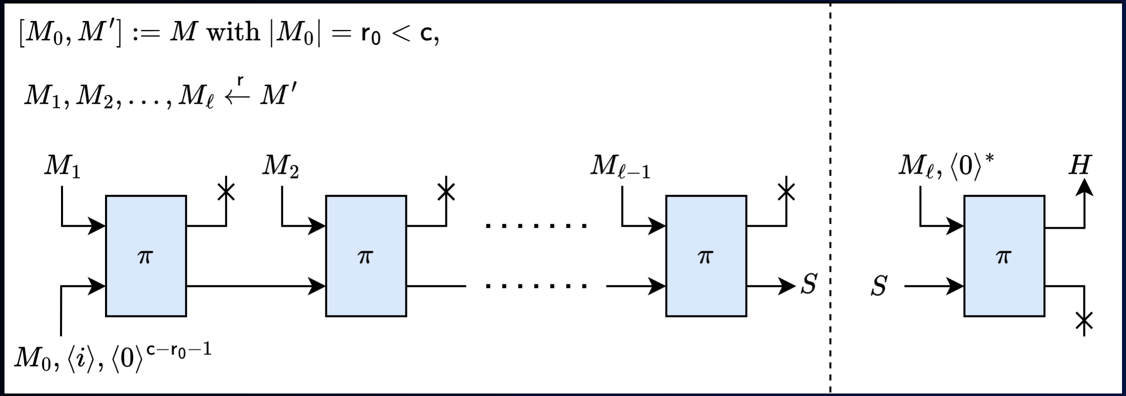
RO-Indifferentiability of GSponge



Sponge2 is RO-Indiff

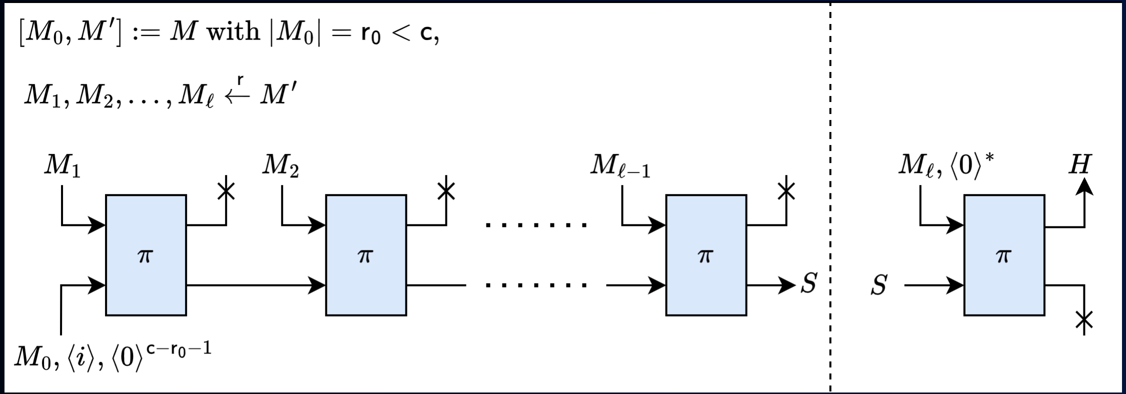
→ GSponge$[u, r_0, p, \mathsf{pad}]$ is RO-Indiff

RO-Indifferentiability of Sponge2



(d) Sponge2 Hashing Mode

RO-Indifferentiability of Sponge2

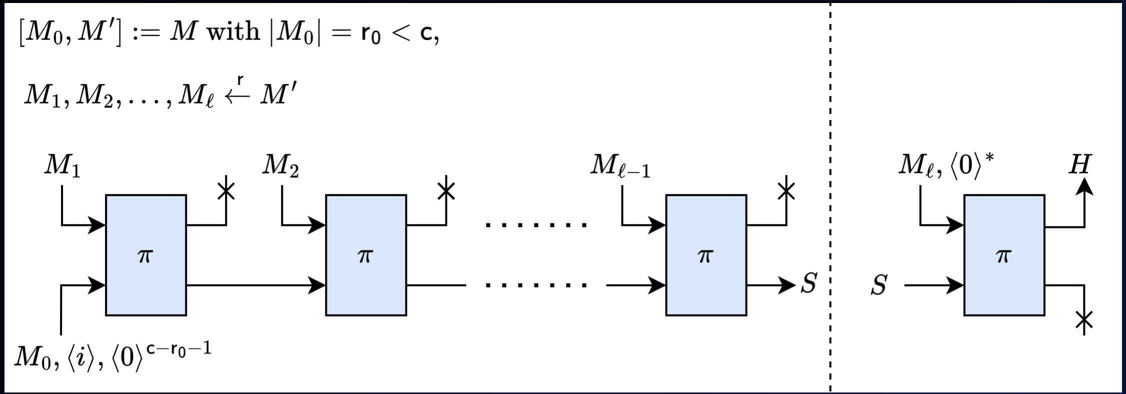


(d) Sponge2 Hashing Mode

- For $\mathsf{r}_0 = \mathsf{c}/2$ and $p > 2$, $\exists\, S$ such that for any $\mathcal{D}$ making q many π calls :

$$\mathcal{D}_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1})) \leq \frac{3q}{p^{\mathsf{c}/2}}$$

RO-Indifferentiability of Sponge2



(d) Sponge2 Hashing Mode

- In security margins, Sponge2 provides $(\mathsf{c} \cdot \log_2 p - 4)/2$ bits of indifferentiability
- i.e., 126 bits of security when $p \approx 2^{64}$ and $\mathsf{c} = 4$

Sponge2: Intuitive Indifferentiability Proof

Capacity Collision Free Functions

- $f^{\mathsf{ccf}} : \mathbb{F}_p^{\mathsf{r+c}} \to \mathbb{F}_p^{\mathsf{r+c}}$ is a capacity-collision-free function (CCF) if

  for i^{th} query $x_i = r_i \| c_i$ with $f^{\mathsf{ccf}}(x_i) = R_i \| C_i$,

$$C_i \notin \{c_1, \ldots, c_i, C_1, \ldots, C_{i-1}\}$$

Capacity Collision Free Functions

- $f^{\mathsf{ccf}} : \mathbb{F}_p^{\mathsf{r}+\mathsf{c}} \to \mathbb{F}_p^{\mathsf{r}+\mathsf{c}}$ is a capacity-collision-free function (CCF) if

  for i^{th} query $x_i = r_i \| c_i$ with $f^{\mathsf{ccf}}(x_i) = R_i \| C_i$,

$$C_i \notin \{c_1, \ldots, c_i, C_1, \ldots, C_{i-1}\}$$

- $(f_1^{\mathsf{ccf}}, f_2^{\mathsf{ccf}})$ is a CCF pair if for (x_i^1, x_i^2),

$$C_i^1 \notin \{c_1^1, \ldots, c_i^1, c_1^2, \ldots, c_i^2, C_1^1, \ldots, C_{i-1}^1, C_1^2, \ldots, C_{i-1}^2\}$$

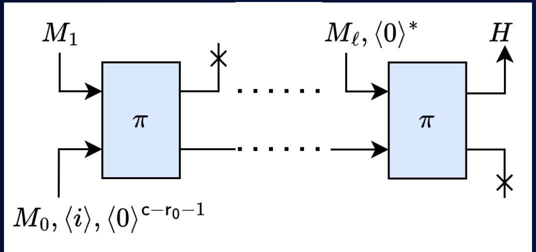
$$C_i^2 \notin \{c_1^1, \ldots, c_i^1, c_1^2, \ldots, c_i^2, C_1^1, \ldots, C_{i-1}^1, C_i^1, C_1^2, \ldots, C_{i-1}^2\}$$

Sponge2: Indifferentiability Proof

Step 1.

- Cross oracle and duplicate queries does not help $\mathcal{D}$
- Let $\mathcal{D}'$ be $\mathcal{D}$ with no cross oracle and duplicate queries

$\mathcal{D}_q((\text{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$



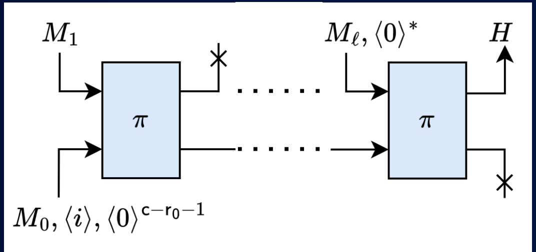
(d) Sponge2 Hashing Mode

Sponge2: Indifferentiability Proof

Step 1.

- Cross oracle and duplicate queries does not help $\mathcal{D}$
- Let $\mathcal{D}'$ be $\mathcal{D}$ with no cross oracle and duplicate queries

$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$



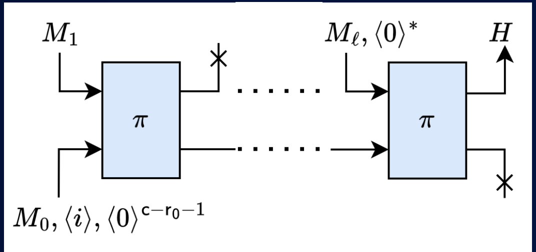
(d) Sponge2 Hashing Mode

Sponge2: Indifferentiability Proof

Step 2.

- Replace permutations with random functions

$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$



(d) Sponge2 Hashing Mode

Sponge2: Indifferentiability Proof

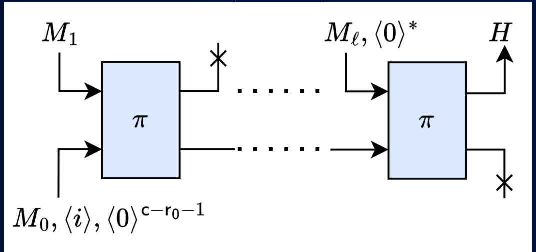
Step 2.

- Replace permutations with random functions

$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$$

$$\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, S, S^{-1}))$$



(d) Sponge2 Hashing Mode

Sponge2: Indifferentiability Proof

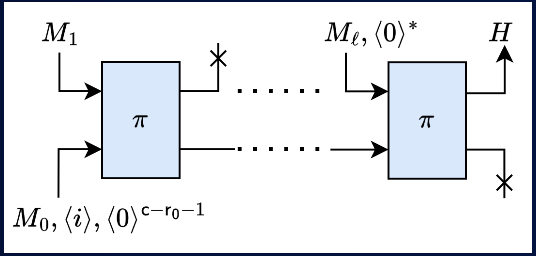
Step 3.

- Redefine random functions with restricted output set $\mathcal{L}$
- Let $g_1, g_2 : \mathbb{F}_p^{\mathsf{r}+\mathsf{c}} \to \mathbb{F}_p^{r+c} \backslash \mathcal{L}$ be uniform random functions

$$\mathcal{D}'_q((\text{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$

$$\Big\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\text{Sponge2}, f_1, f_2), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$



(d) Sponge2 Hashing Mode

Sponge2: Indifferentiability Proof

Step 3.

- Redefine random functions with restricted output set $\mathcal{L}$
- Let $g_1, g_2 : \mathbb{F}_p^{\mathsf{r}+\mathsf{c}} \to \mathbb{F}_p^{r+c} \backslash \mathcal{L}$ be uniform random functions

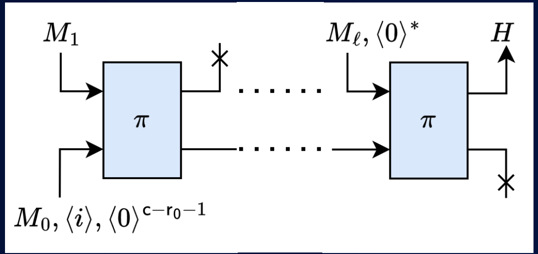
$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$

$$\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$

$$\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, g_1, g_2), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$



(d) Sponge2 Hashing Mode

Sponge2: Indifferentiability Proof

Step 4.

- Replacing g_1, g_2 with their Capacity-collision-free variants $f_1^{\mathsf{ccf}}, f_2^{\mathsf{ccf}}$

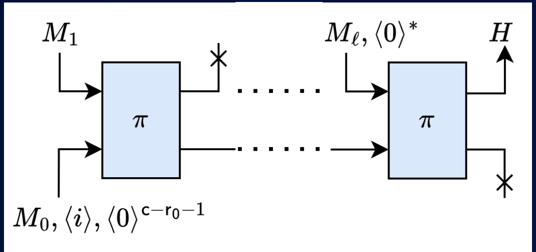
$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$$

$$\Big\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, S, S^{-1}))$$

$$\Big\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, g_1, g_2), (\mathcal{RO}, S, S^{-1}))$$

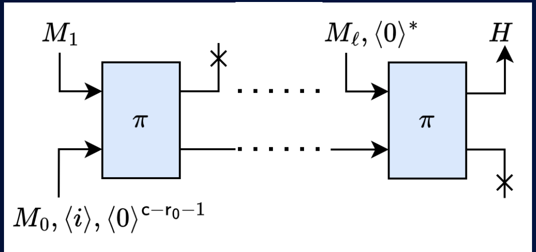


(d) Sponge2 Hashing Mode

Sponge2: Indifferentiability Proof

Step 4.

- Replacing g_1, g_2 with their Capacity-collision-free variants $f_1^{\mathsf{ccf}}, f_2^{\mathsf{ccf}}$



(d) Sponge2 Hashing Mode

$$\mathcal{D}'_q((\text{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$$

$$\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\text{Sponge2}, f_1, f_2), (\mathcal{RO}, S, S^{-1}))$$

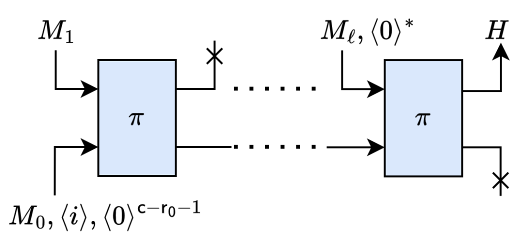
$$\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\text{Sponge2}, g_1, g_2), (\mathcal{RO}, S, S^{-1})) \xrightarrow{+ \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}} \mathcal{D}'_q((\text{Sponge2}, f_1^{\mathsf{ccf}}, f_2^{\mathsf{ccf}}), (\mathcal{RO}, S, S^{-1}))$$

Sponge2: Indifferentiability Proof

Step 5.

- Define Sponge2′ by pulling the post-padded 0s to the start of message



(d) Sponge2 Hashing Mode

$$\mathcal{D}'_q((\text{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$$

$$\downarrow + \frac{q(q-1)}{2p^{r+c}}$$

$$\mathcal{D}'_q((\text{Sponge2}, f_1, f_2), (\mathcal{RO}, S, S^{-1}))$$

$$\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{r+c}}$$

$$\mathcal{D}'_q((\text{Sponge2}, g_1, g_2), (\mathcal{RO}, S, S^{-1})) \xrightarrow{+ \frac{q^2}{p^c - |\mathcal{L}|p^{-r}}} \mathcal{D}'_q((\text{Sponge2}, f_1^{ccf}, f_2^{ccf}), (\mathcal{RO}, S, S^{-1}))$$

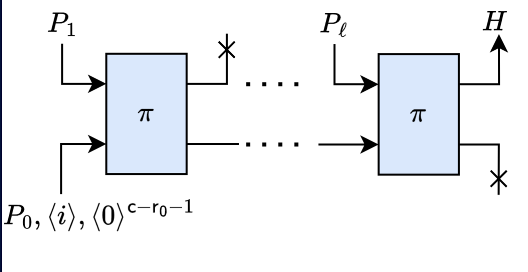
Sponge2: Indifferentiability Proof

Step 5.

- Define $\mathsf{Sponge2}'$ by pulling the post-padded 0s to the start of message

$[P_0, M'] := [\langle 0\rangle^*, M]$ with $|P_0| = \mathsf{r}_0 < \mathsf{c}$,

$P_1, P_2, \ldots, P_\ell \xleftarrow{\mathsf{r}} M'$



(d) $\mathsf{Sponge2}'$ Hashing Mode

$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$

$$\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$

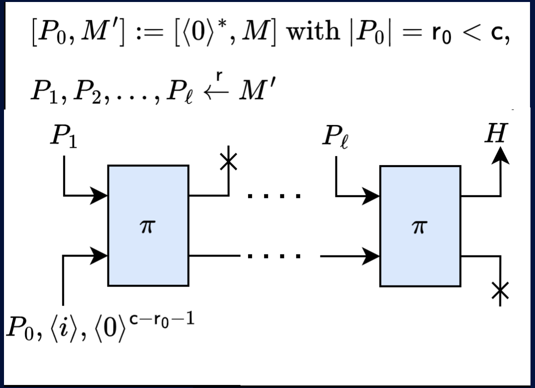
$$\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, g_1, g_2), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1})) \xrightarrow{+ \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}} \mathcal{D}'_q((\mathsf{Sponge2}, f_1^{\mathsf{ccf}}, f_2^{\mathsf{ccf}}), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$

Sponge2: Indifferentiability Proof

Step 5.

- Define Sponge2$'$ by pulling the post-padded 0s to the start of message



(d) Sponge2$'$Hashing Mode

$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$

$$\Big\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$

$$\Big\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, g_1, g_2), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1})) \xrightarrow{+ \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}} \mathcal{D}'_q((\mathsf{Sponge2}', f_1^{\mathsf{ccf}}, f_2^{\mathsf{ccf}}), (\mathcal{RO}, \mathcal{S}, \mathcal{S}^{-1}))$$

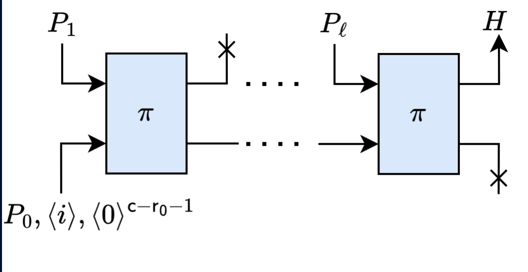
Sponge2: Indifferentiability Proof

Step 6.

- Set $\mathcal{L}$ as the set of all valid first primitive call's inputs and set $S^{-1} = f_2^{\mathsf{ccf}}$

$[P_0, M'] := [\langle 0\rangle^*, M]$ with $|P_0| = \mathsf{r}_0 < \mathsf{c}$,

$P_1, P_2, \ldots, P_\ell \xleftarrow{\mathsf{r}} M'$



(d) Sponge2′ Hashing Mode

$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$$

$$\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, S, S^{-1}))$$

$$\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, g_1, g_2), (\mathcal{RO}, S, S^{-1})) \xrightarrow{+ \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}} \mathcal{D}'_q((\mathsf{Sponge2}', f_1^{\mathsf{ccf}}, f_2^{\mathsf{ccf}}), (\mathcal{RO}, S, S^{-1}))$$

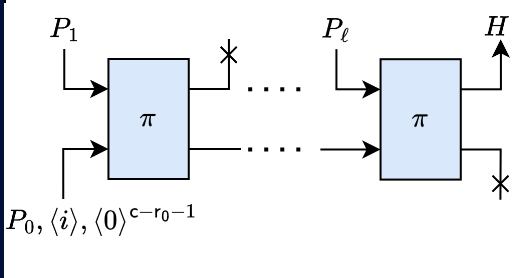
Sponge2: Indifferentiability Proof

Step 6.

- Set $\mathcal{L}$ as the set of all valid first primitive call's inputs and set $S^{-1} = f_2^{\mathsf{ccf}}$

$[P_0, M'] := [\langle 0 \rangle^*, M]$ with $|P_0| = \mathsf{r}_0 < \mathsf{c}$,

$P_1, P_2, \ldots, P_\ell \xleftarrow{\mathsf{r}} M'$



(d) Sponge2′Hashing Mode

$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$$

$$\Big\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, S, S^{-1}))$$

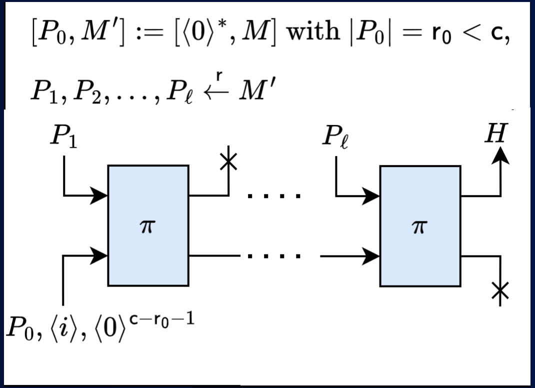
$$\Big\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, g_1, g_2), (\mathcal{RO}, S, S^{-1})) \xrightarrow{+ \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}} \mathcal{D}'_q((\mathsf{Sponge2}', f_1^{\mathsf{ccf}}, f_2^{\mathsf{ccf}}), (\mathcal{RO}, S, f_2^{\mathsf{ccf}}))$$

Sponge2: Indifferentiability Proof

Step 6.

- Set $\mathcal{L}$ as the set of all valid first primitive call's inputs and set $S^{-1} = f_2^{\mathsf{ccf}}$



(d) Sponge2′Hashing Mode

$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$$

$$\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, S, S^{-1}))$$

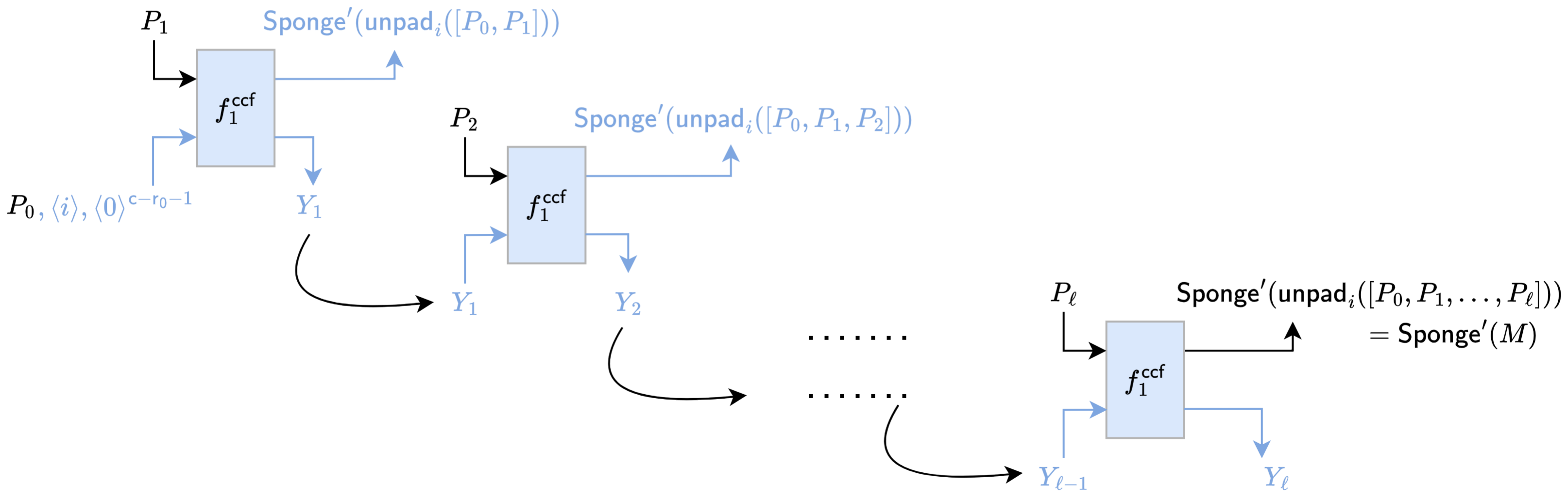
$$\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, g_1, g_2), (\mathcal{RO}, S, S^{-1})) \xrightarrow{+ \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}| p^{-\mathsf{r}}}} \mathcal{D}'_q((\mathsf{Sponge2}', f_1^{\mathsf{ccf}}), (\mathcal{RO}, S))$$

Sponge2: Indifferentiability Proof

$[P_0, M'] := [\langle 0\rangle^*, M]$ with $|M_0| = \mathsf{r}_0 < \mathsf{c}$,

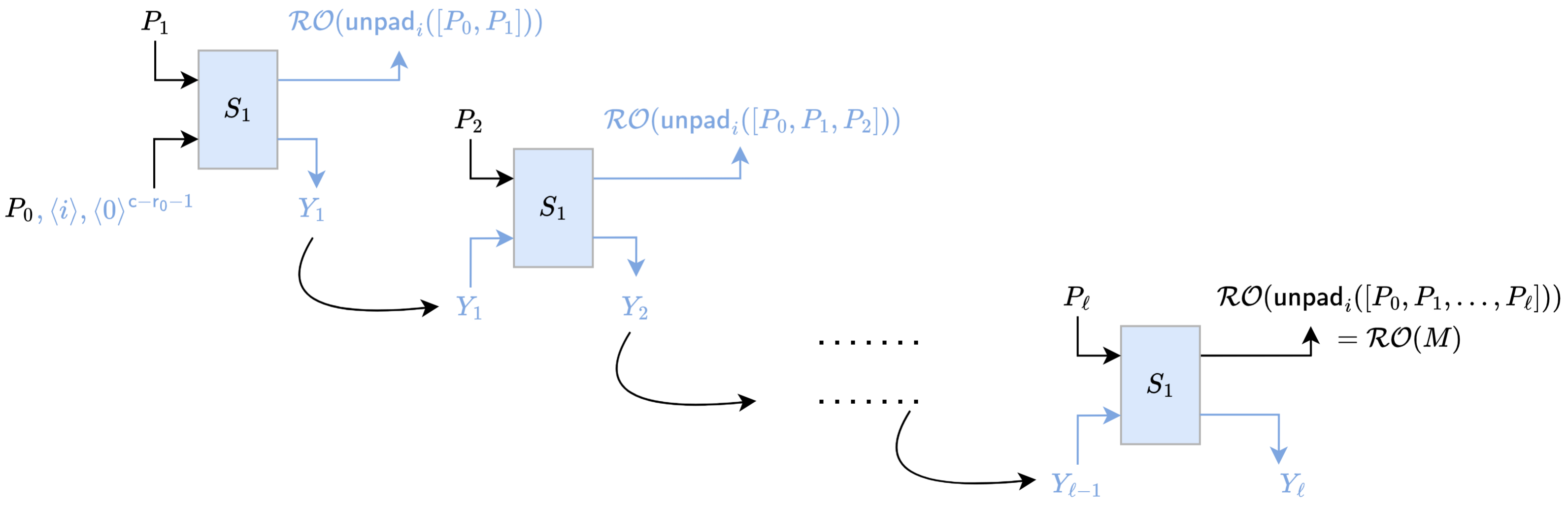
$P_1, P_2, \ldots, P_\ell \xleftarrow{\mathsf{r}} M'$



Sponge2: Indifferentiability Proof

$[P_0, M'] := [\langle 0\rangle^*, M]$ with $|M_0| = \mathsf{r}_0 < \mathsf{c}$,

$P_1, P_2, \ldots, P_\ell \xleftarrow{\mathsf{r}} M'$



Sponge2: Indifferentiability Proof

Since f_1^{ccf} is a random CCF :

- $\mathsf{Sponge2}'$ outputs are uniformly distributed
- S is a random CCF
- S is consistent with $\mathcal{RO}$

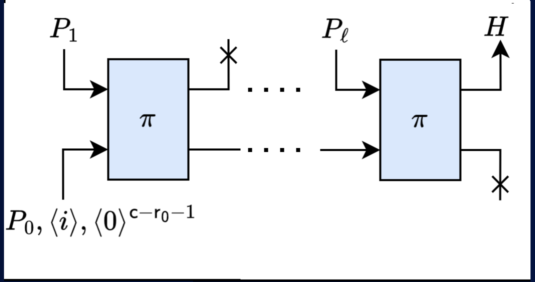
Sponge2: Indifferentiability Proof

Step 6.

- Set $\mathcal{L}$ as the set of all valid first primitive call's inputs and set $S^{-1} = f_2^{\mathsf{ccf}}$

$[P_0, M'] := [\langle 0\rangle^*, M]$ with $|P_0| = \mathsf{r}_0 < \mathsf{c}$,

$P_1, P_2, \ldots, P_\ell \xleftarrow{\mathsf{r}} M'$



(d) Sponge2′Hashing Mode

$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$$

$$\Big\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, S, S^{-1}))$$

$$\Big\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, g_1, g_2), (\mathcal{RO}, S, S^{-1})) \xrightarrow{+ \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}} \mathcal{D}'_q((\mathsf{Sponge2}', f_1^{\mathsf{ccf}}), (\mathcal{RO}, S))$$

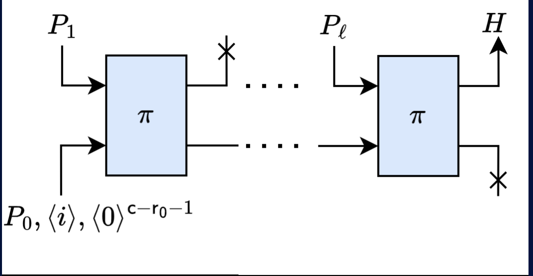
Sponge2: Indifferentiability Proof

Step 6.

- Set $\mathcal{L}$ as the set of all valid first primitive call's inputs and set $S^{-1} = f_2^{\mathsf{ccf}}$

$$[P_0, M'] := [\langle 0\rangle^*, M] \text{ with } |P_0| = \mathsf{r}_0 < \mathsf{c},$$

$$P_1, P_2, \ldots, P_\ell \xleftarrow{\mathsf{r}} M'$$



(d) Sponge2′Hashing Mode

$$\mathcal{D}'_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1}))$$

$$\downarrow + \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, f_1, f_2), (\mathcal{RO}, S, S^{-1}))$$

$$\downarrow + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}}$$

$$\mathcal{D}'_q((\mathsf{Sponge2}, g_1, g_2), (\mathcal{RO}, S, S^{-1})) \xrightarrow{+ \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}} \mathcal{D}'_q((\mathsf{Sponge2}', f_1^{\mathsf{ccf}}), (\mathcal{RO}, S)) = 0$$

Sponge2: Indifferentiability Proof

- Combine all steps

$$\mathcal{D}_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1})) \leq \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}} + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}} + \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}| p^{-\mathsf{r}}}$$

Sponge2: Indifferentiability Proof

- Combine all steps with $|\mathcal{L}| < 3p^{\mathsf{r}+\mathsf{r}_0}/2$, $\mathsf{r}_0 = \mathsf{c}/2$ and $p > 2$.

$$\mathcal{D}_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1})) \leq \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}} + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}} + \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}$$

Sponge2: Indifferentiability Proof

- Combine all steps with $|\mathcal{L}| < 3p^{\mathsf{r}+\mathsf{r}_0}/2$, $\mathsf{r}_0 = \mathsf{c}/2$ and $p > 2$.

$$\mathcal{D}_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1})) \leq \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}} + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}} + \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}$$

$$\leq \frac{3q}{p^{\mathsf{c}/2}}$$

Sponge2: Indifferentiability Proof

- Combine all steps with $|\mathcal{L}| < 3p^{\mathsf{r}+\mathsf{r}_0}/2$, $\mathsf{r}_0 = \mathsf{c}/2$ and $p > 2$.

$$\mathcal{D}_q((\mathsf{Sponge2}, \pi, \pi^{-1}), (\mathcal{RO}, S, S^{-1})) \leq \frac{q(q-1)}{2p^{\mathsf{r}+\mathsf{c}}} + q \cdot \frac{|\mathcal{L}|}{p^{\mathsf{r}+\mathsf{c}}} + \frac{q^2}{p^{\mathsf{c}} - |\mathcal{L}|p^{-\mathsf{r}}}$$

$$\leq \frac{3q}{p^{\mathsf{c}/2}}$$

- *Capacity-collision-free functions* (CCFs) ➔ simpler Indifferentiability proofs with no security degradation!!

Open Questions

Design Problems:

- Q.1 Are there better CCF designs than public permutations and functions?

Open Questions

Design Problems:

- Q.1 Are there better CCF designs than public permutations and functions?

Provable Security and Applications:

- Q.2 Can CCFs simplify quantum security analysis for Sponges?
- Q.3 What can be other applications of CCFs than Sponges?

Open Questions

Design Problems:

- Q.1 Are there better CCF designs than public permutations and functions?

more efficient and secure

Provable Security and Applications:

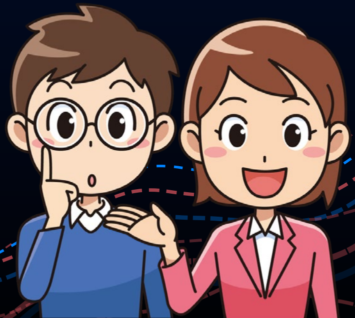
- Q.2 Can CCFs simplify quantum security analysis for Sponges?

- Q.3 What can be other applications of CCFs than Sponges?

Thank You!

(ia.cr/2024/911)

Contact: Amitsingh.bhati@3milabs.tech



Template Credits: Slidesgo and Freepik