Key-Homomorphic and Aggregate Verifiable Random Functions

G. Malavolta (Bocconi University)

European Research Council

Established by the European Commission

https://eprint.iacr.org/2024/643.pdf

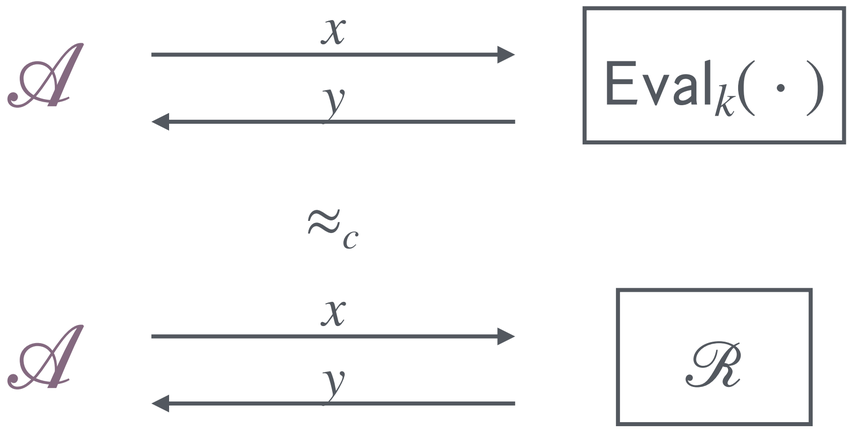
Verifiable Random Functions [MRV99]

$$\mathscr{F} : \mathscr{K} \times \{0,1\}^{\lambda} \to \mathscr{Y}$$

Verifiable Random Functions [MRV99]

$$\mathcal{F} : \mathcal{K} \times \{0,1\}^{\lambda} \to \mathcal{Y}$$

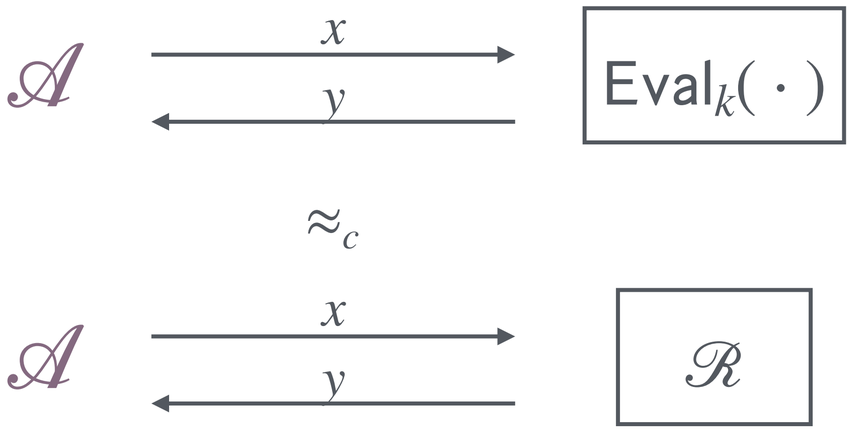
PSEUDORANDOMNESS



Verifiable Random Functions [MRV99]

$$\mathcal{F} : \mathcal{K} \times \{0,1\}^{\lambda} \to \mathcal{Y}$$

PSEUDORANDOMNESS



UNIQUENESS

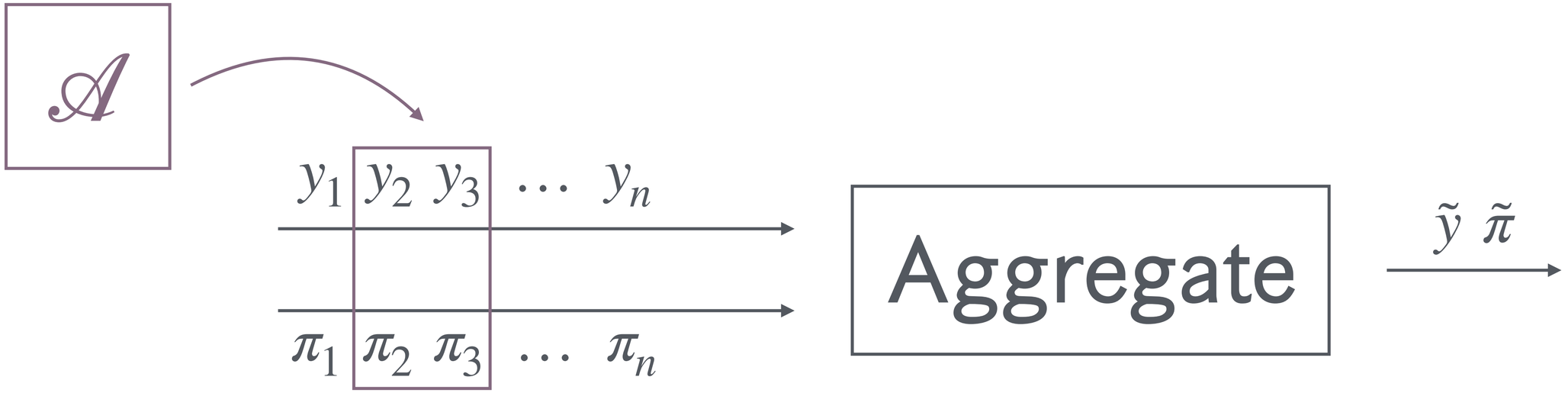
$\mathsf{Prove}_k(x) \to \pi$

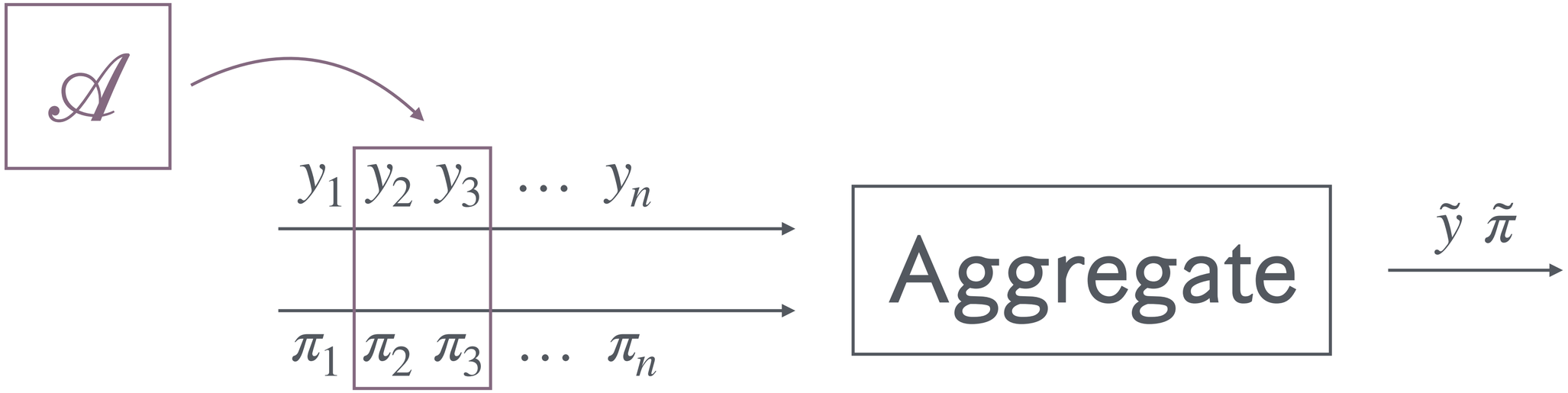
$\mathsf{Verify}(x, y, \pi) \to \{0,1\}$

$$\Pr_{\mathcal{A}}\left[\mathsf{Verify}(x^*, y^*, \pi^*) = 1 : y^* \neq \mathsf{Eval}_k(x^*)\right] \approx 0$$

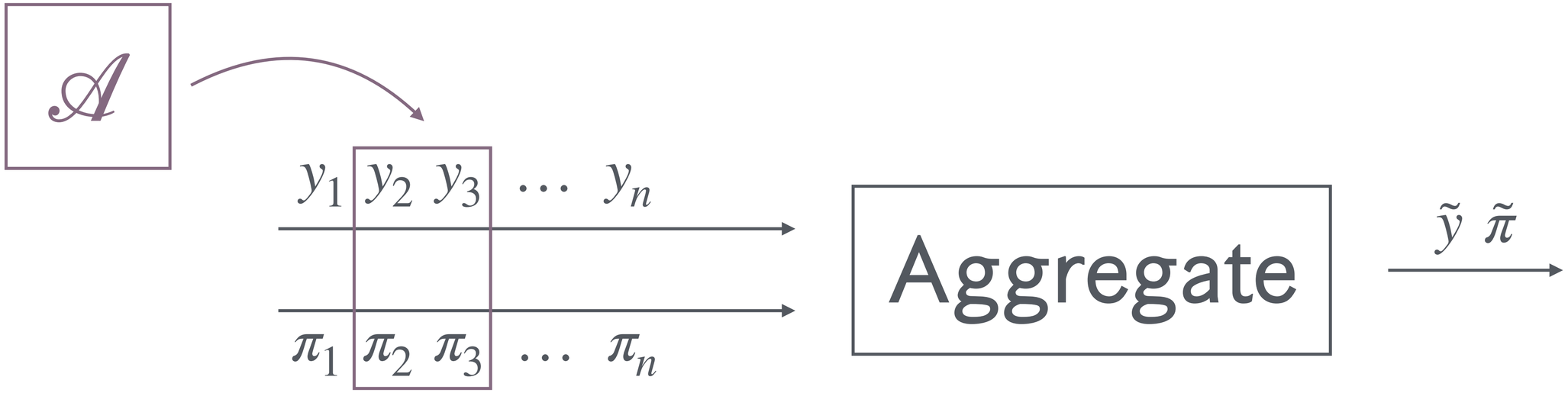
$(x^*, y^*, \pi^*) \leftarrow \mathcal{A}$


$y_1\ y_2\ y_3\ \cdots\ y_n$
$\pi_1\ \pi_2\ \pi_3\ \ldots\ \pi_n$
Aggregate
$\tilde{y}\ \tilde{\pi}$

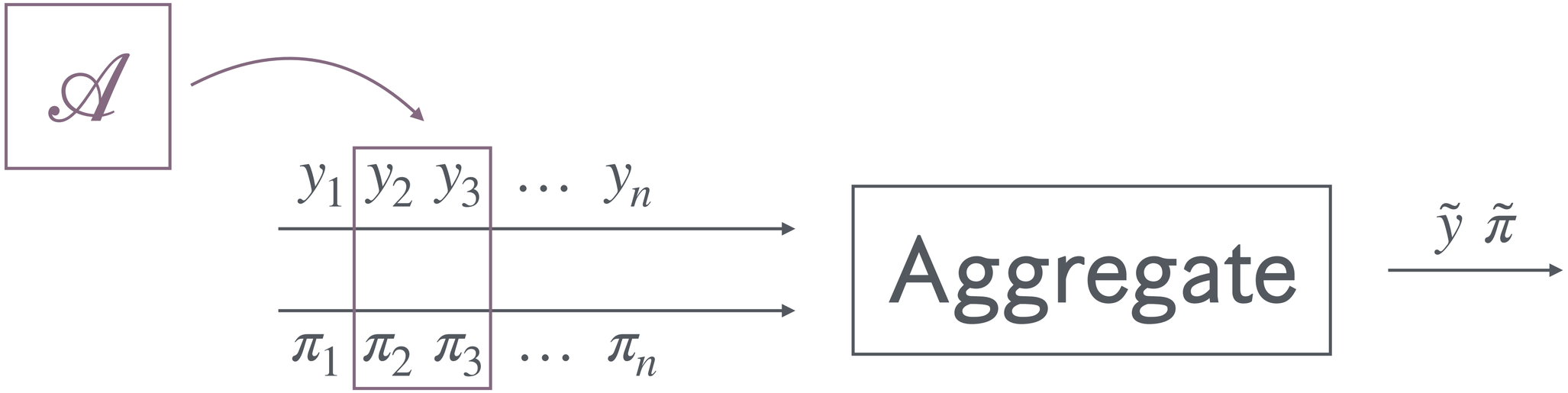

$\mathscr{A}$
y_1 y_2 y_3 … y_n
π_1 π_2 π_3 … π_n
Aggregate
$\tilde{y}$ $\tilde{\pi}$



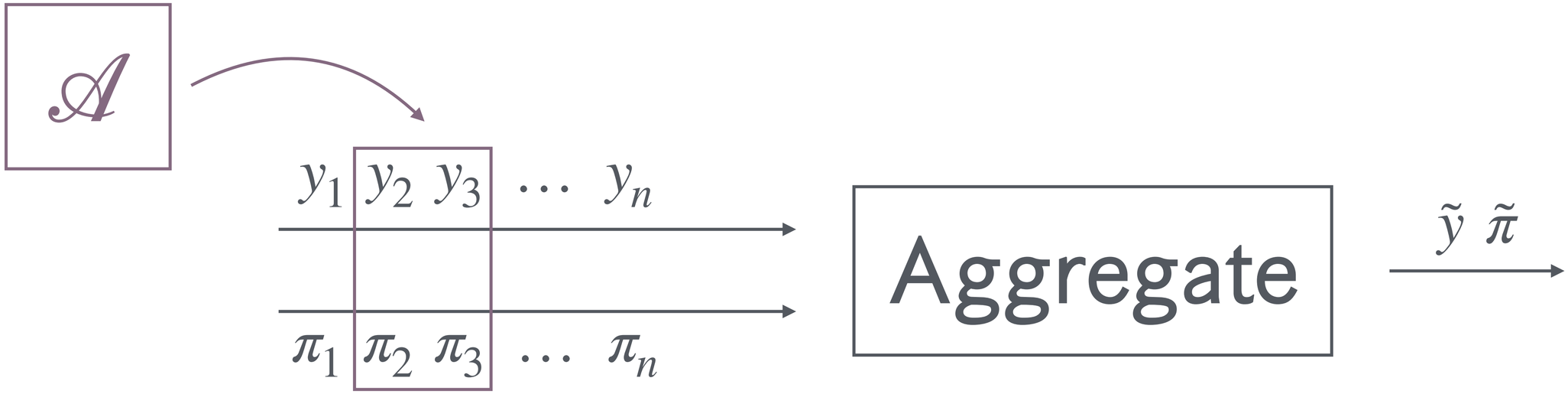
- Well-studied for unpredictable functions (= digital signatures)



- Well-studied for unpredictable functions (= digital signatures)
- Ideal aggregate signature



- Well-studied for unpredictable functions (= digital signatures)
- Ideal aggregate signature
- Simple one-round distributed VRF



- Well-studied for unpredictable functions (= digital signatures)
- Ideal aggregate signature
- Simple one-round distributed VRF
- Many other conceivable applications

Results

Results

- Definitions

Results

- Definitions
  - (Aggregate) Pseudorandomness

Results

- Definitions
  - (Aggregate) Pseudorandomness
  - (Aggregate) Uniqueness

Results

- Definitions
  - (Aggregate) Pseudorandomness
  - (Aggregate) Uniqueness
  - (Aggregate) Binding

Results

- Definitions
  - (Aggregate) Pseudorandomness
  - (Aggregate) Uniqueness
  - (Aggregate) Binding
- Pairing-based construction

Results

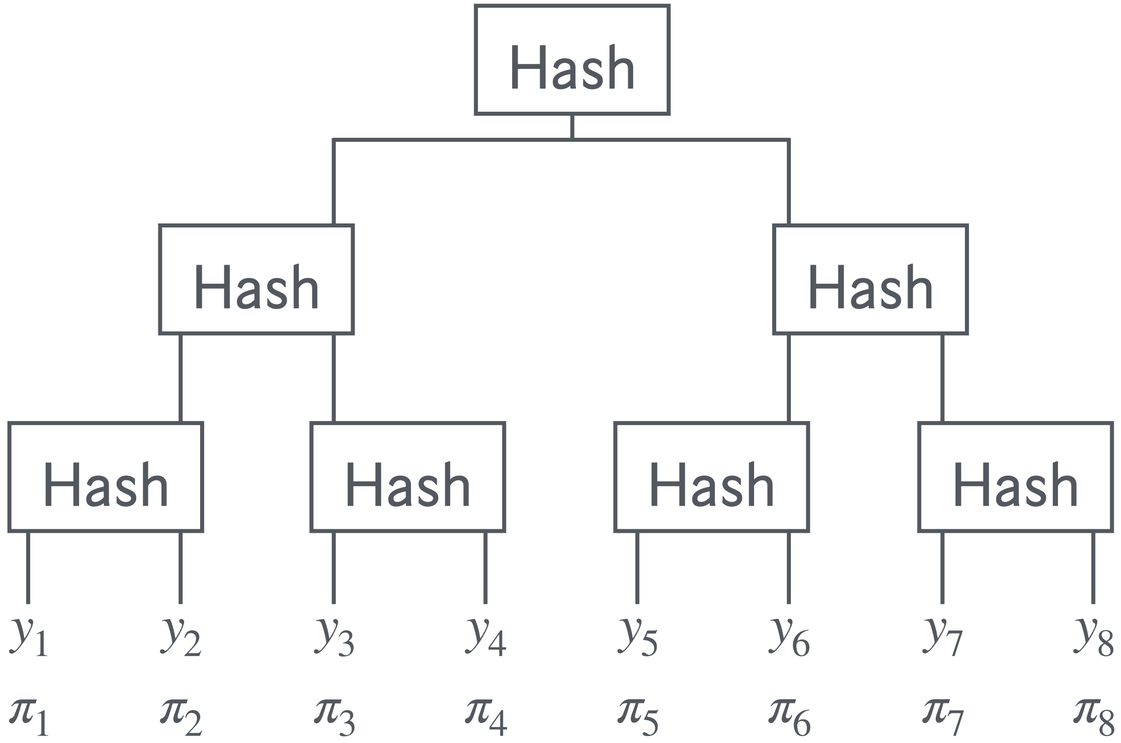
- Definitions
  - (Aggregate) Pseudorandomness
  - (Aggregate) Uniqueness
  - (Aggregate) Binding
- Pairing-based construction
  - Concretely efficient, key-homomorphic, ROM

Results

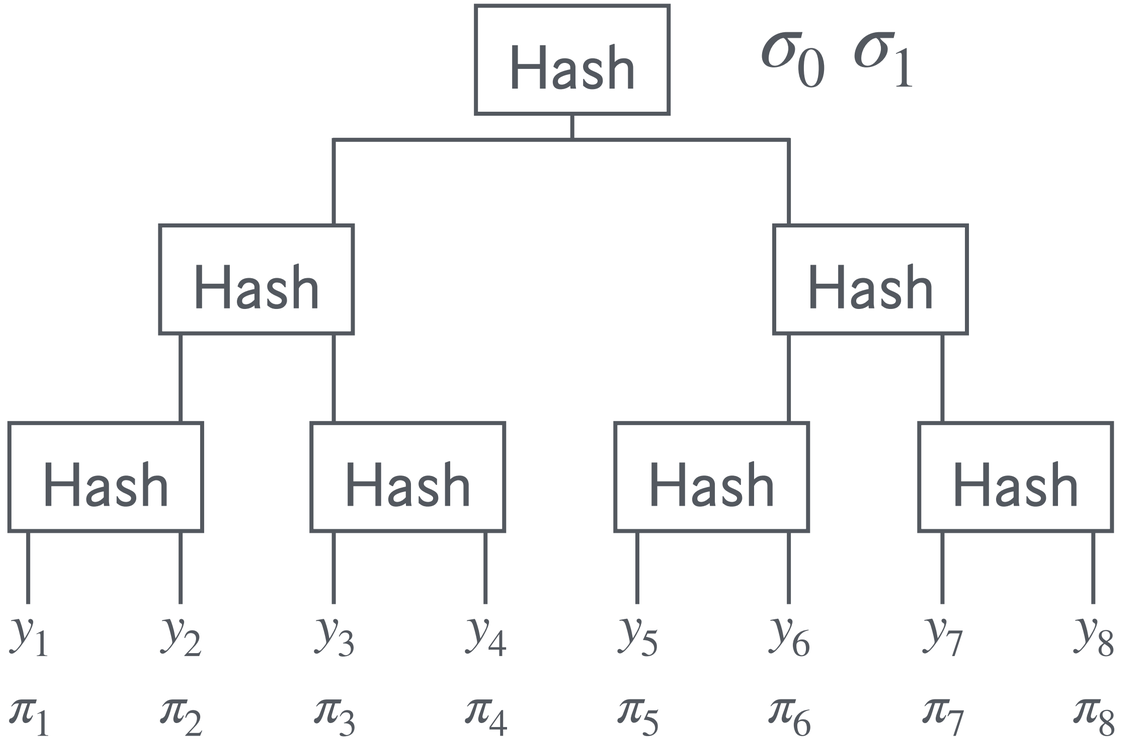
- Definitions
  - (Aggregate) Pseudorandomness
  - (Aggregate) Uniqueness
  - (Aggregate) Binding
- Pairing-based construction
  - Concretely efficient, key-homomorphic, ROM
- Generic (LWE-based) construction

Results

- Definitions
  - (Aggregate) Pseudorandomness
  - (Aggregate) Uniqueness
  - (Aggregate) Binding
- Pairing-based construction
  - Concretely efficient, key-homomorphic, ROM
- Generic (LWE-based) construction

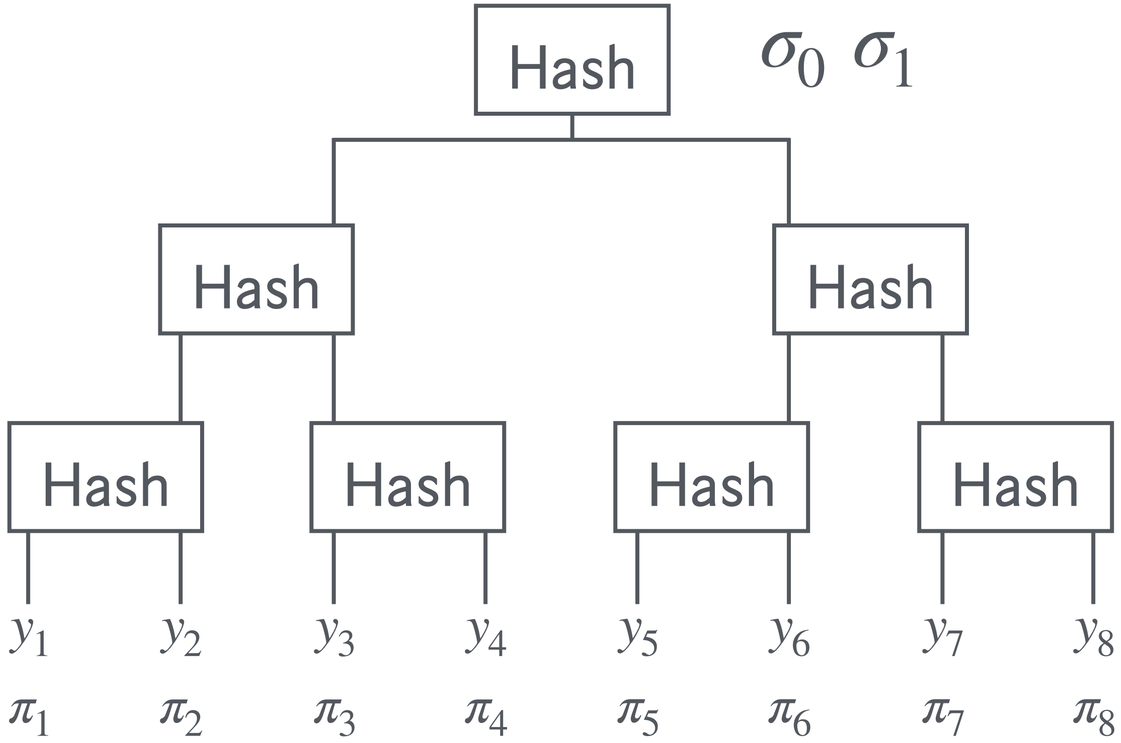

Hash
Hash
Hash
Hash
Hash
Hash
Hash
y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
π_1 π_2 π_3 π_4 π_5 π_6 π_7 π_8

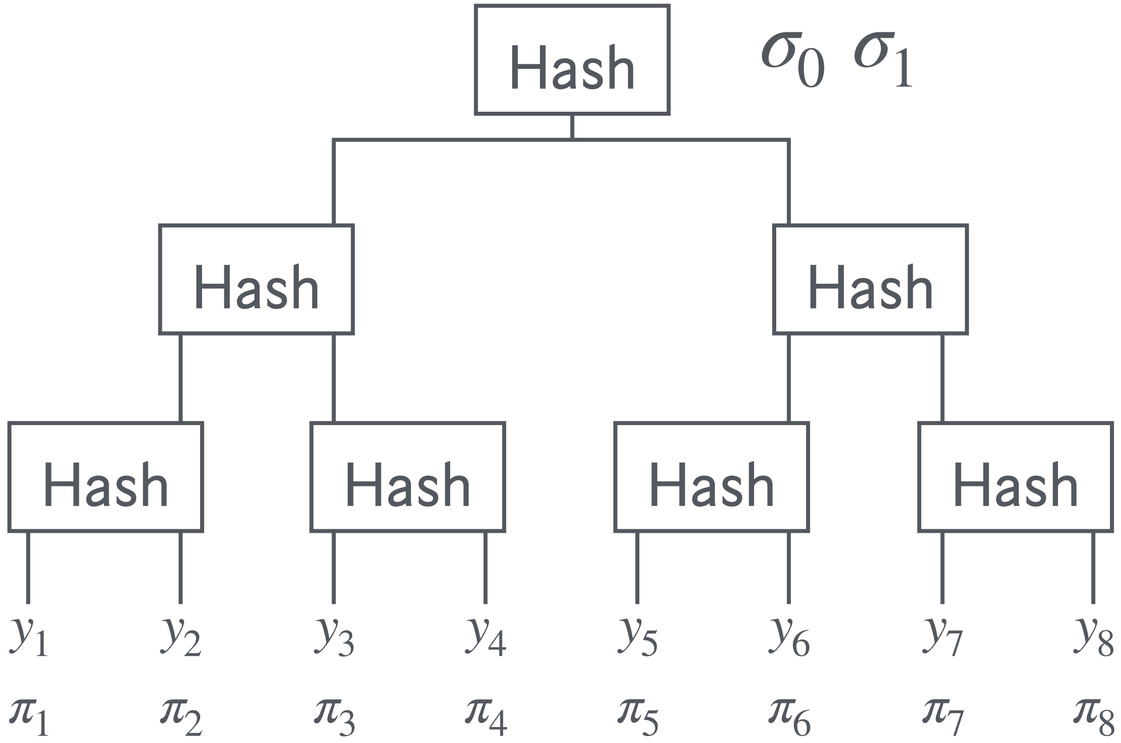
BATCH ARGUMENTS FOR NP



BATCH ARGUMENTS FOR NP

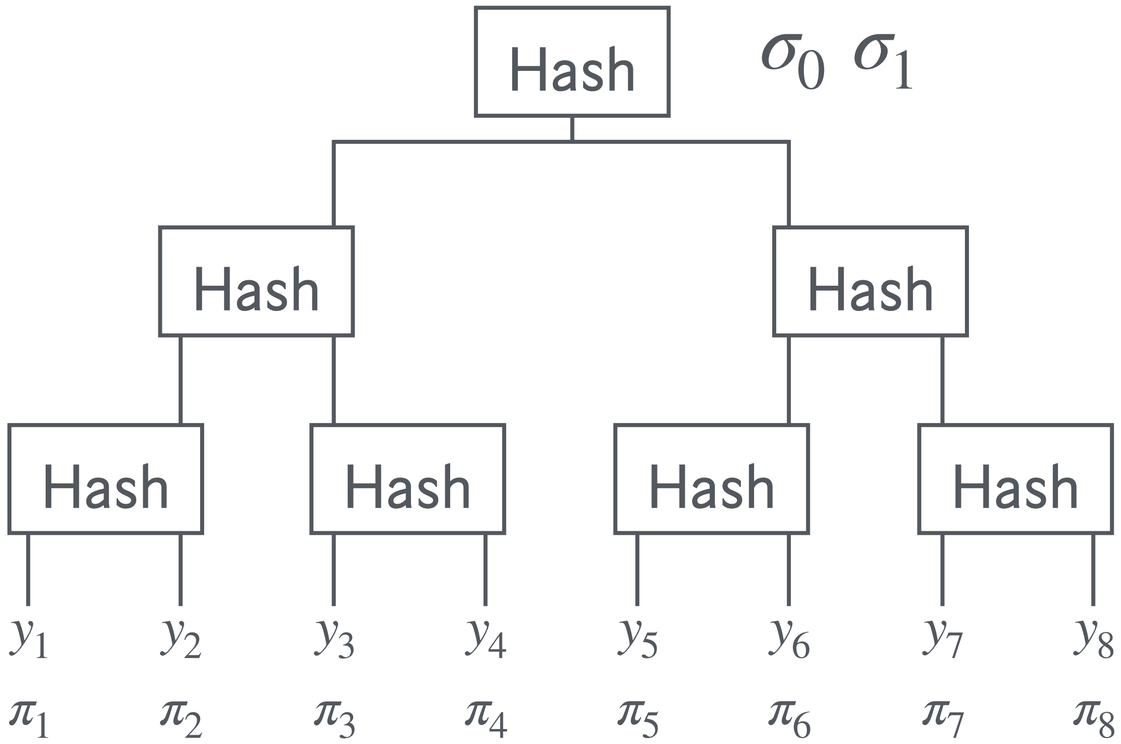
- Compute two BARGs for the same statement:





BATCH ARGUMENTS FOR NP

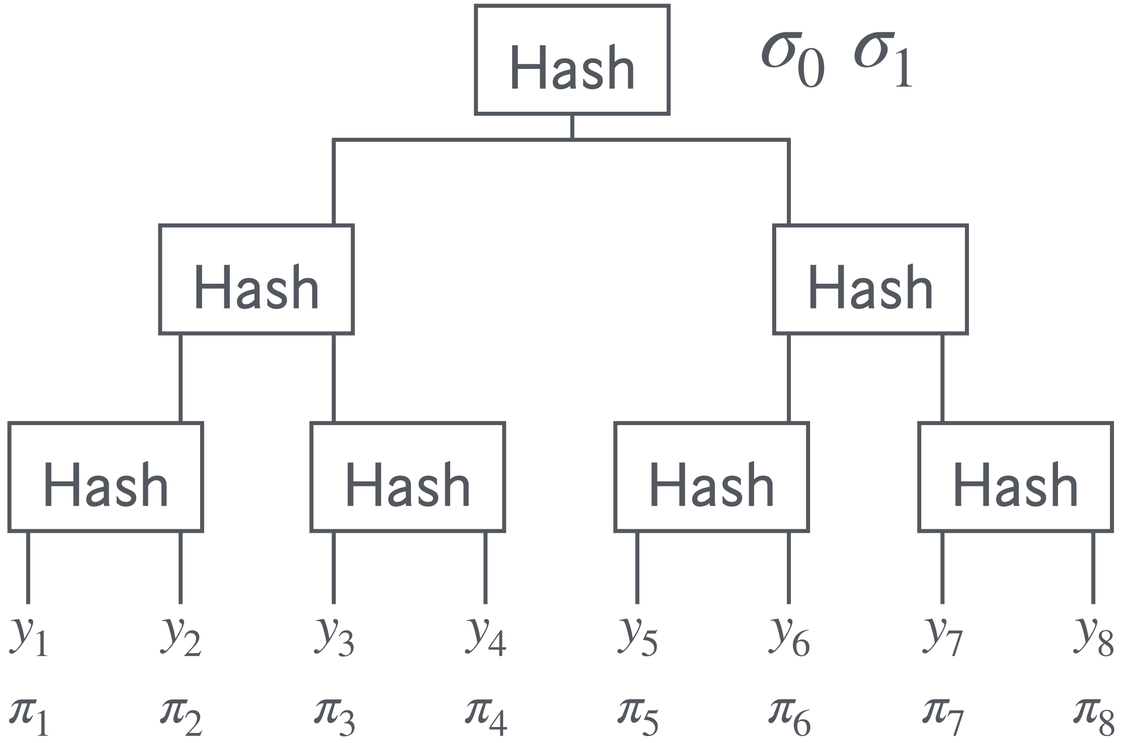
- Compute two BARGs for the same statement:
  - For all indices i, there is a tuple (π_i, y_i, p_i) s.t.



BATCH ARGUMENTS FOR NP

- Compute two BARGs for the same statement:
  - For all indices i, there is a tuple (π_i, y_i, p_i) s.t.

    A. p_i is a valid root-to-leaf path

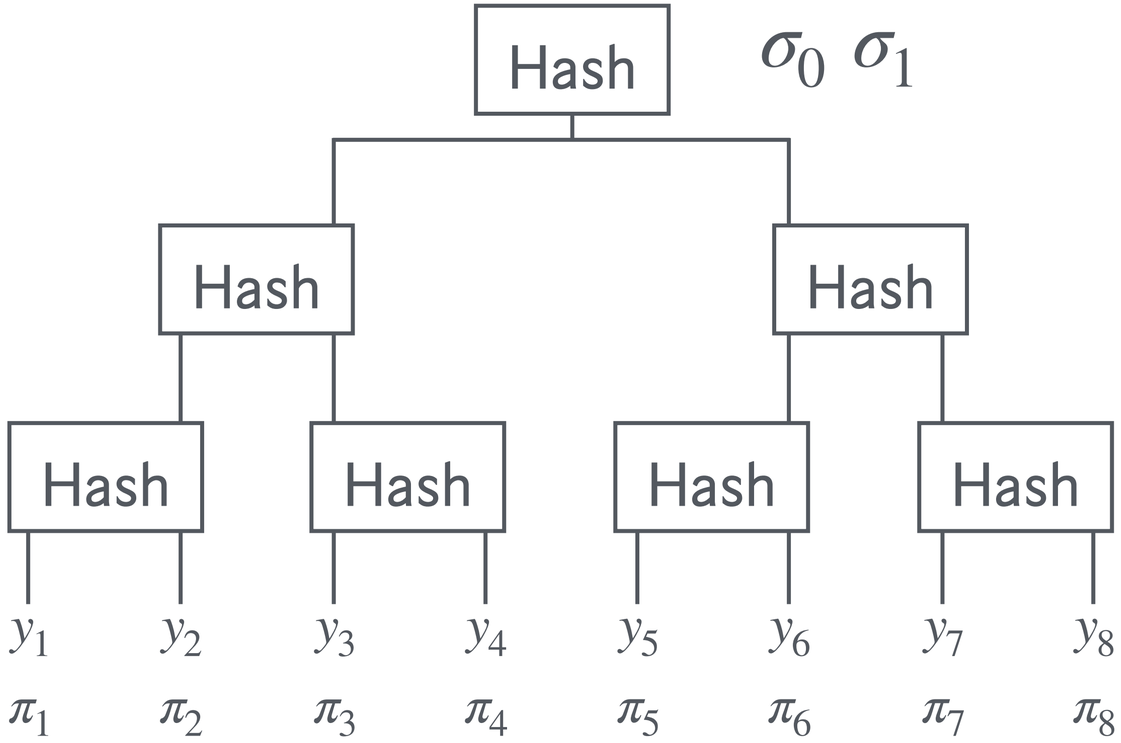


BATCH ARGUMENTS FOR NP

- Compute two BARGs for the same statement:
  - For all indices i, there is a tuple (π_i, y_i, p_i) s.t.

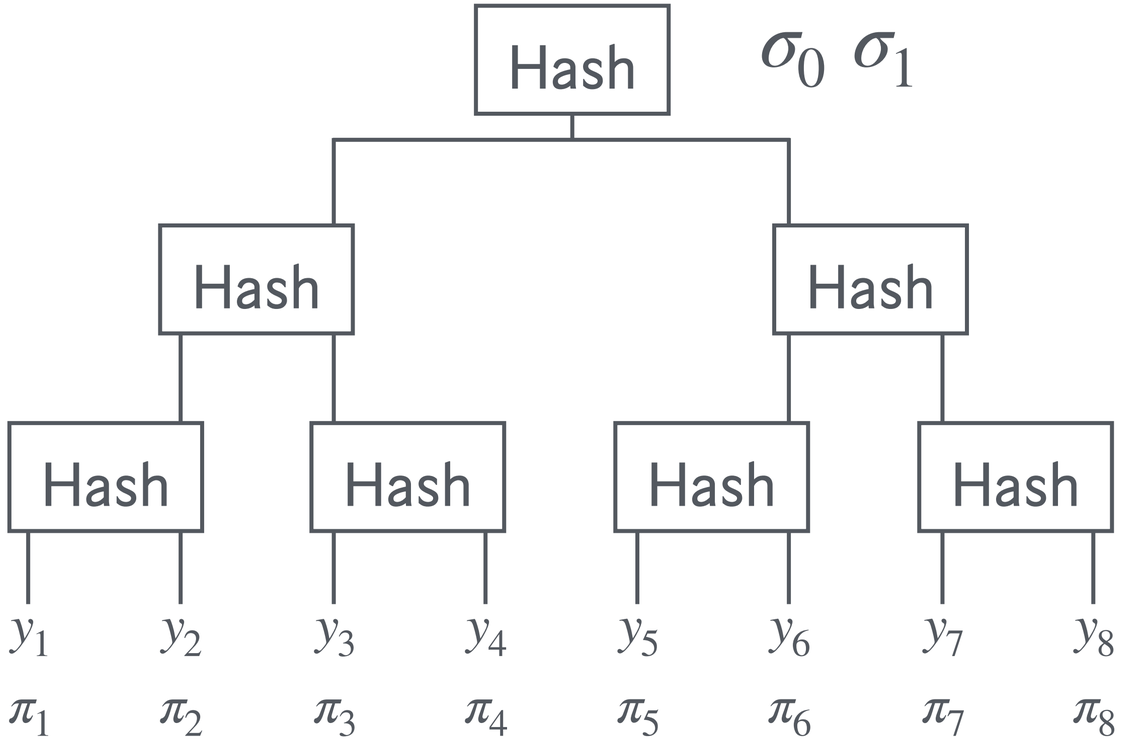
    A. p_i is a valid root-to-leaf path

    B. π_i is a valid proof for y_i



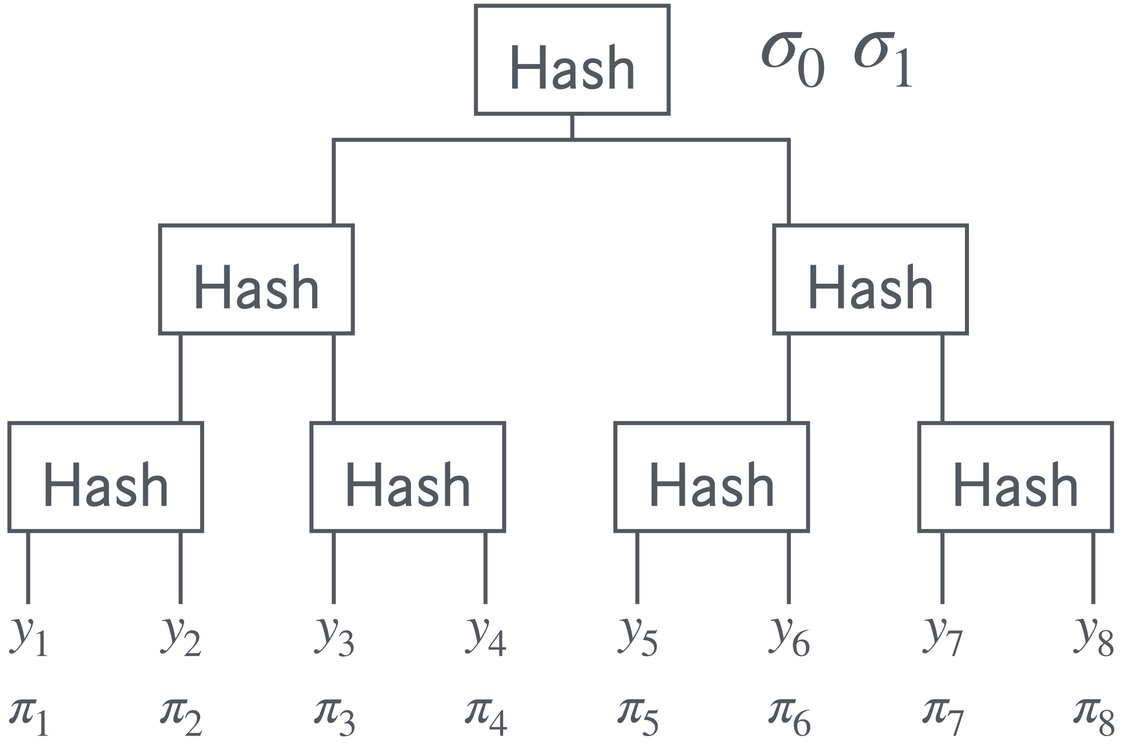
BATCH ARGUMENTS FOR NP

- Compute two BARGs for the same statement:
  - For all indices i, there is a tuple (π_i, y_i, p_i) s.t.
    A. p_i is a valid root-to-leaf path
    B. π_i is a valid proof for y_i
- We require the BARGs to be:



BATCH ARGUMENTS FOR NP

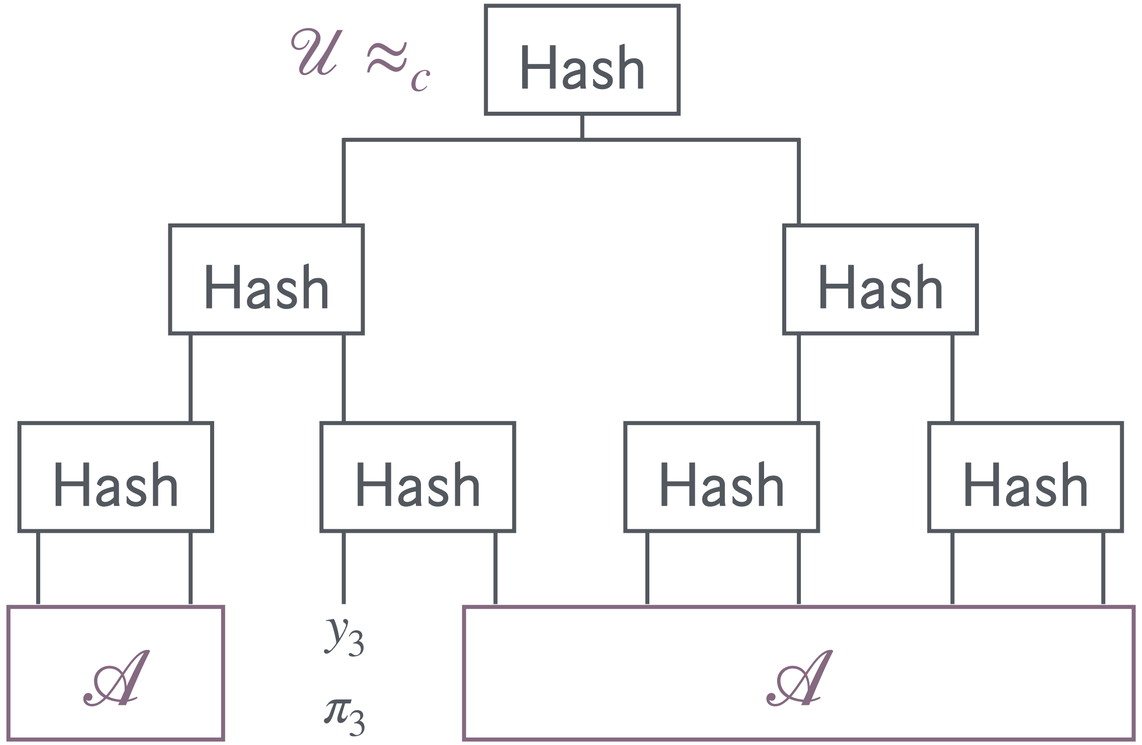
- Compute two BARGs for the same statement:
  - For all indices i, there is a tuple (π_i, y_i, p_i) s.t.
    A. p_i is a valid root-to-leaf path
    B. π_i is a valid proof for y_i
- We require the BARGs to be:
  1. Somewhere extractable



BATCH ARGUMENTS FOR NP

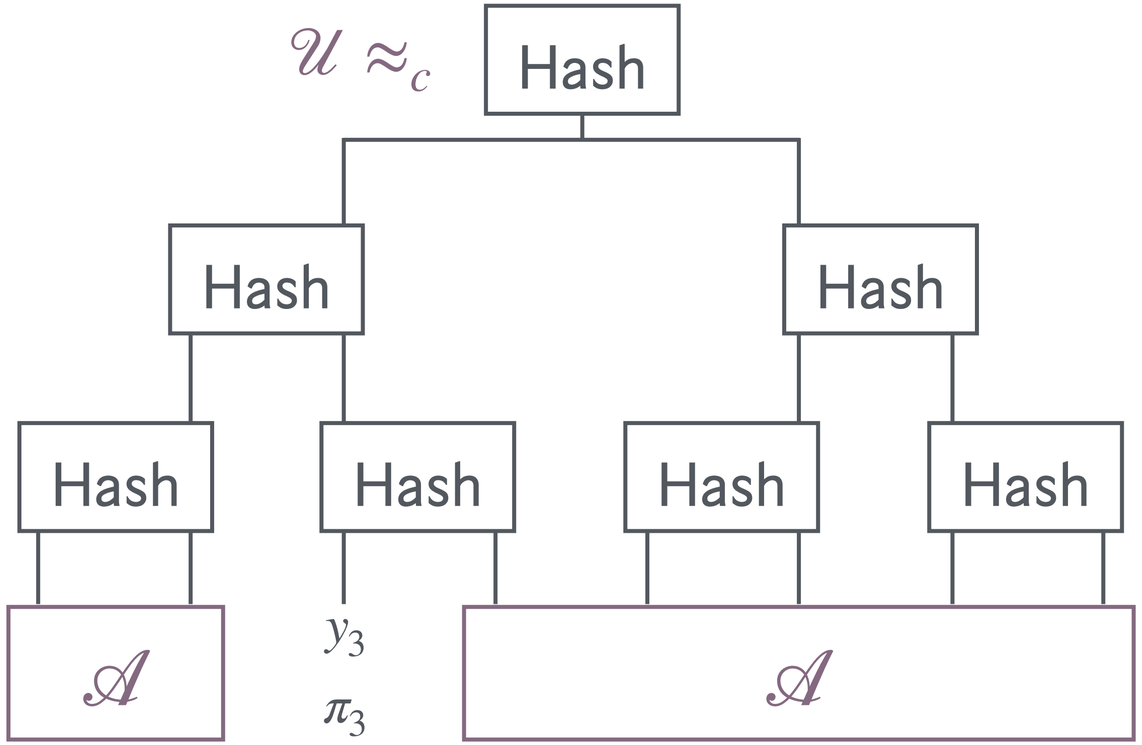
- Compute two BARGs for the same statement:
  - For all indices i, there is a tuple (π_i, y_i, p_i) s.t.
    A. p_i is a valid root-to-leaf path
    B. π_i is a valid proof for y_i
- We require the BARGs to be:
  1. Somewhere extractable
  2. Index-hiding

(AGGREGATE) PSEUDORANDOMNESS



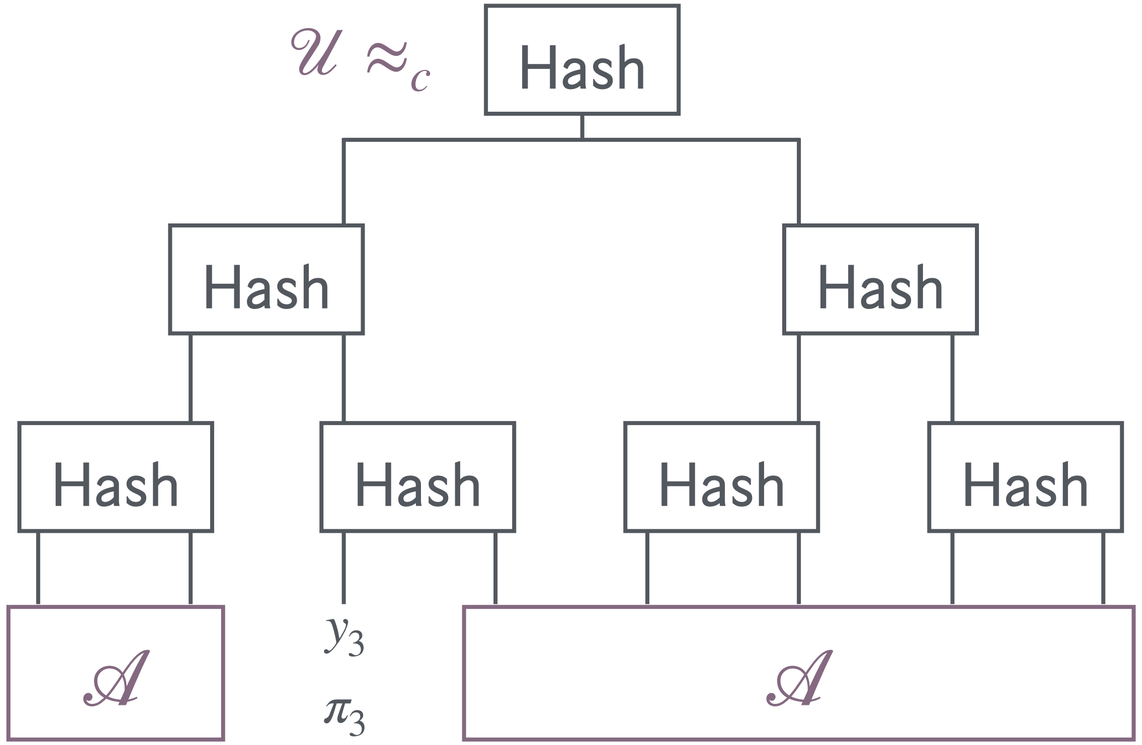
(AGGREGATE) PSEUDORANDOMNESS

- Want: Adversary cannot bias the distribution



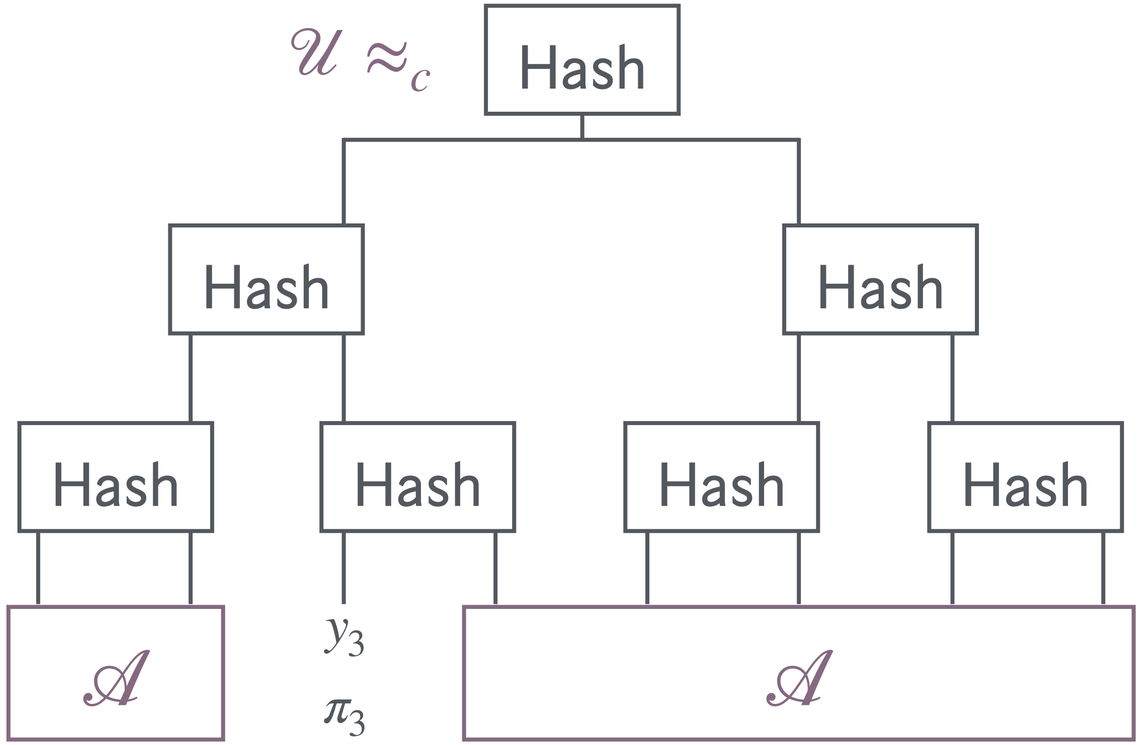
(AGGREGATE) PSEUDORANDOMNESS

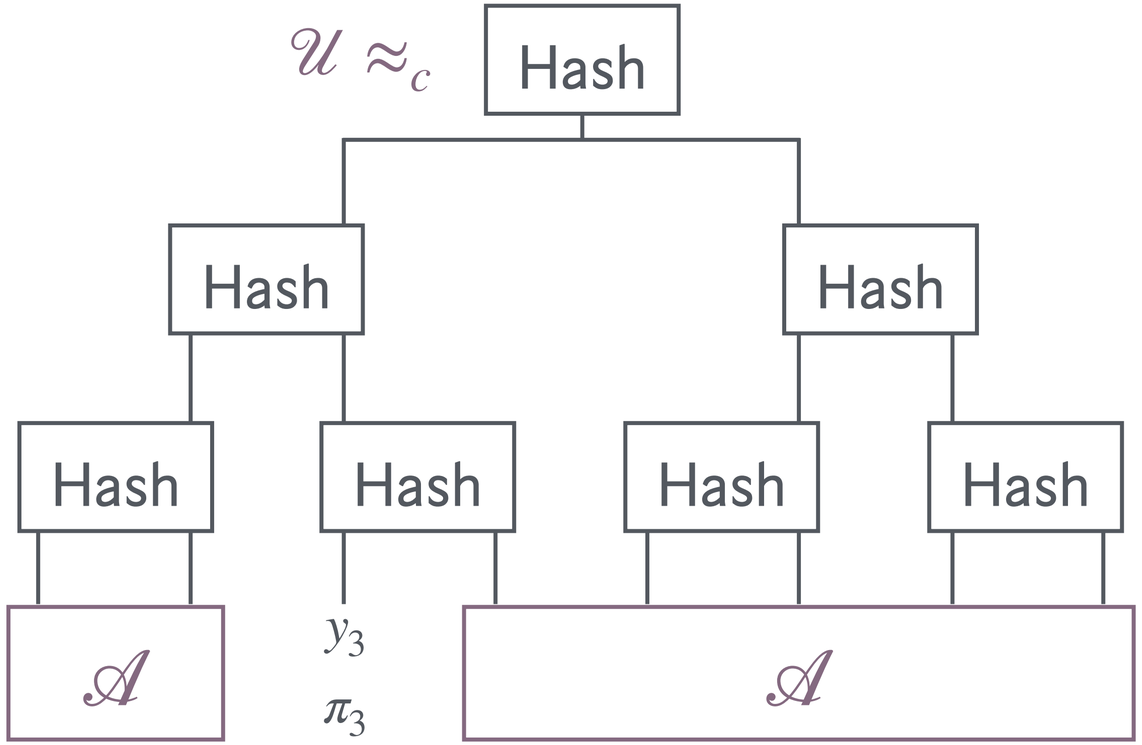
- Want: Adversary cannot bias the distribution
- Idea I: Function-binding hash



(AGGREGATE) PSEUDORANDOMNESS

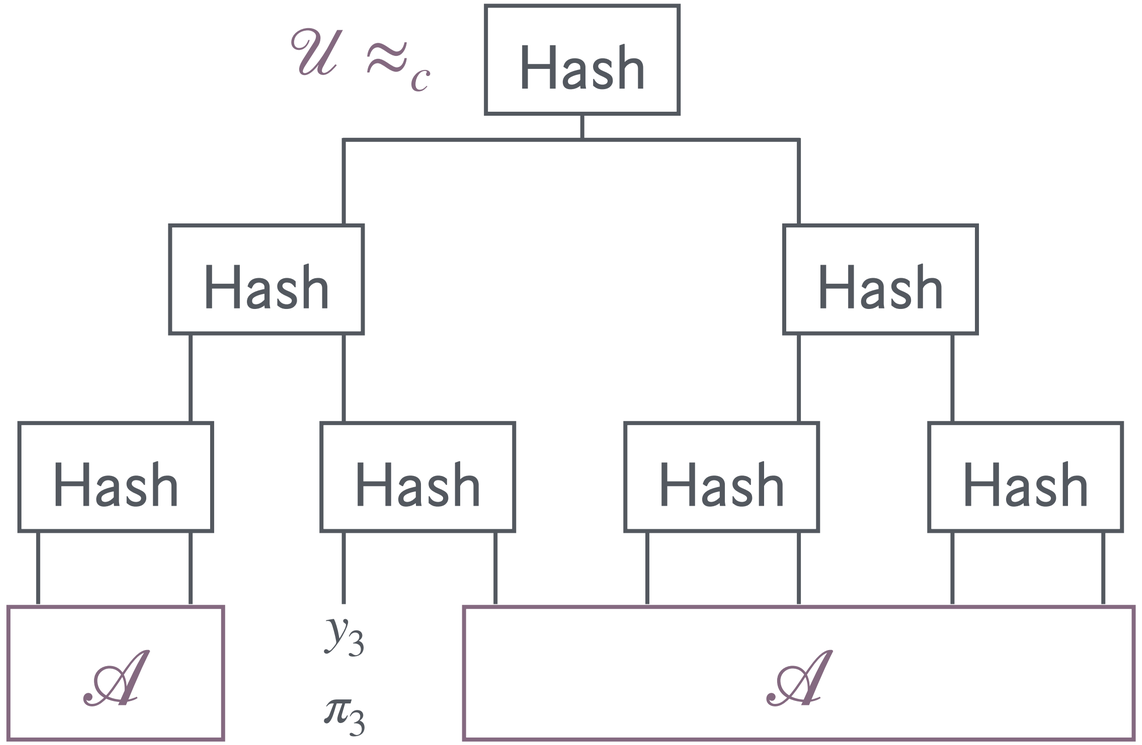
- Want: Adversary cannot bias the distribution
- Idea I: Function-binding hash
  - Make the root statistically binding wrt any node of the tree





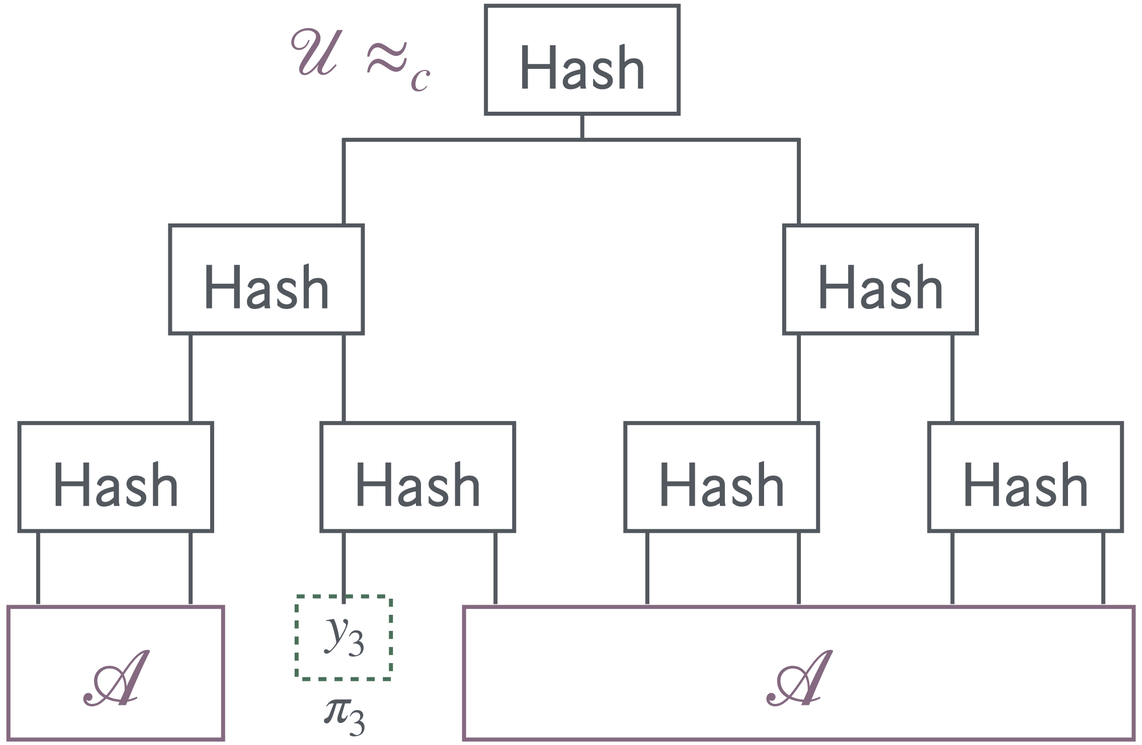
(AGGREGATE) PSEUDORANDOMNESS

- Want: Adversary cannot bias the distribution
- Idea I: Function-binding hash
  - Make the root statistically binding wrt any node of the tree
  - The target node is computationally hidden



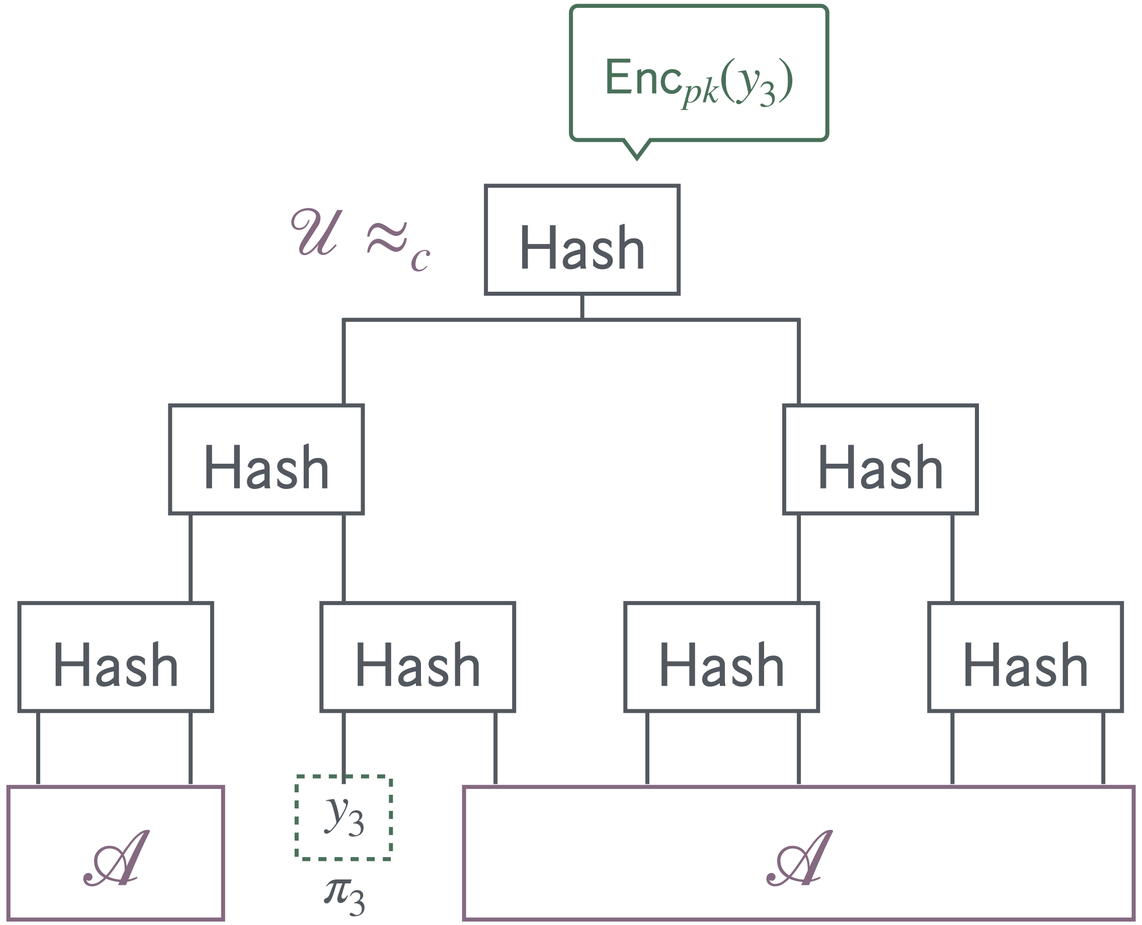
(AGGREGATE) PSEUDORANDOMNESS

- Want: Adversary cannot bias the distribution
- Idea I: Function-binding hash
  - Make the root statistically binding wrt any node of the tree
  - The target node is computationally hidden
  - Construction from rate-1 FHE [HW15, BDGM19, GH19]



(AGGREGATE) PSEUDORANDOMNESS

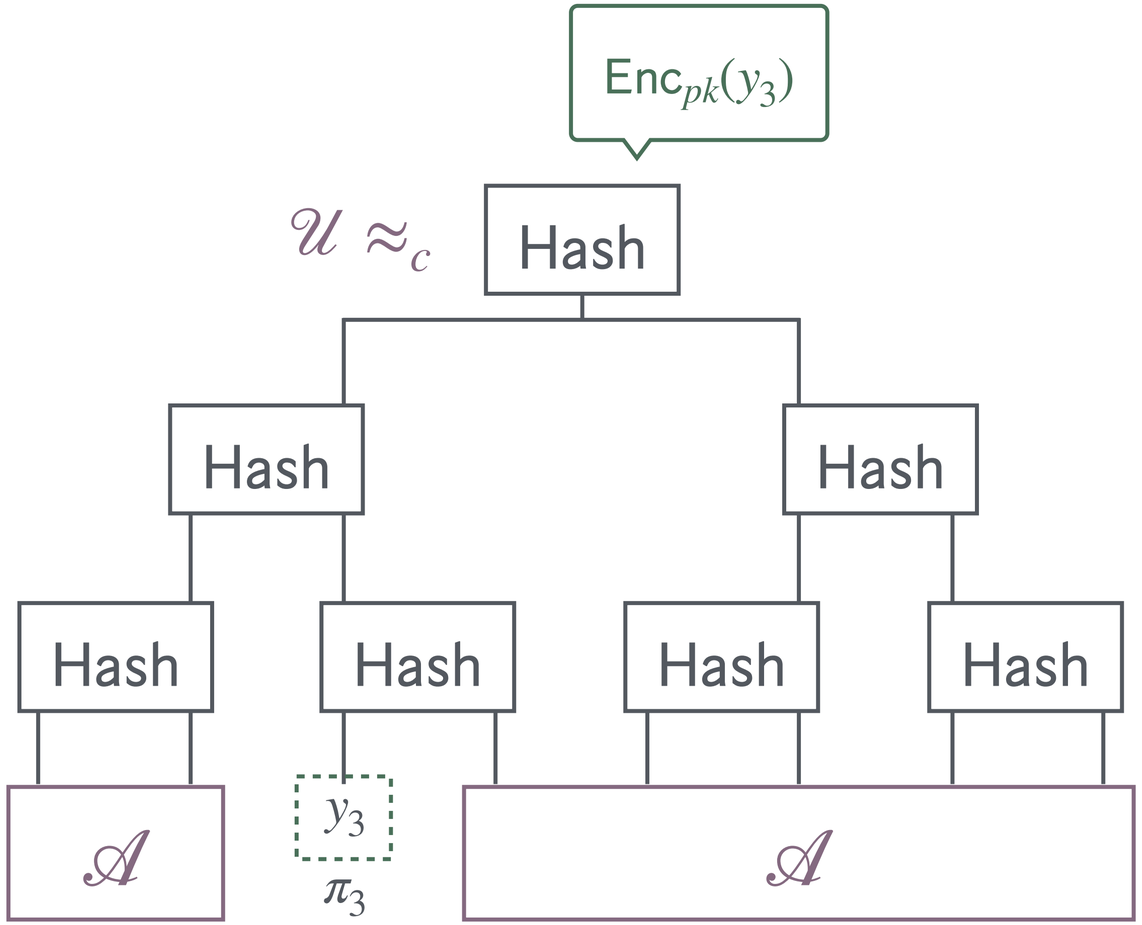
- Want: Adversary cannot bias the distribution
- Idea I: Function-binding hash
  - Make the root statistically binding wrt any node of the tree
  - The target node is computationally hidden
  - Construction from rate-1 FHE [HW15, BDGM19, GH19]



(AGGREGATE) PSEUDORANDOMNESS

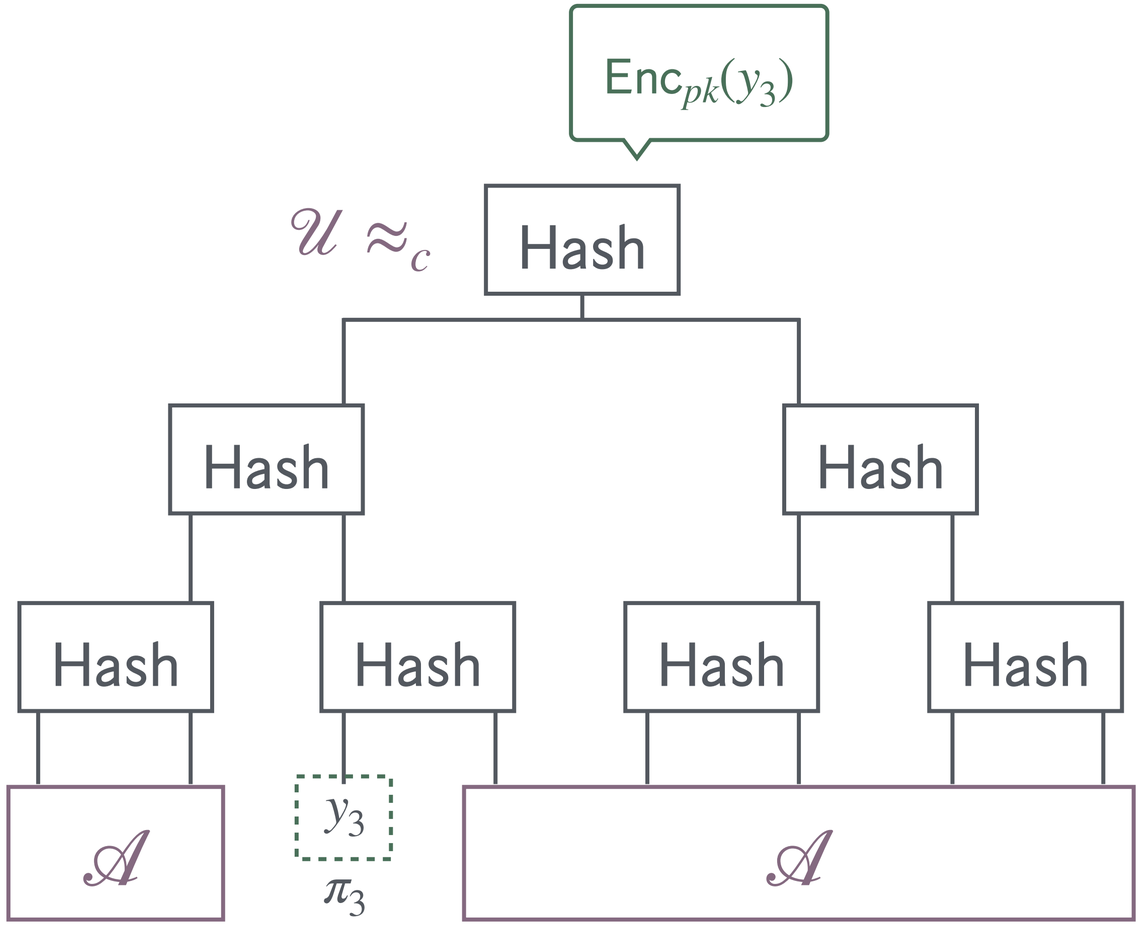
- Want: Adversary cannot bias the distribution
- Idea I: Function-binding hash
  - Make the root statistically binding wrt any node of the tree
  - The target node is computationally hidden
  - Construction from rate-1 FHE [HW15, BDGM19, GH19]

(AGGREGATE) PSEUDORANDOMNESS

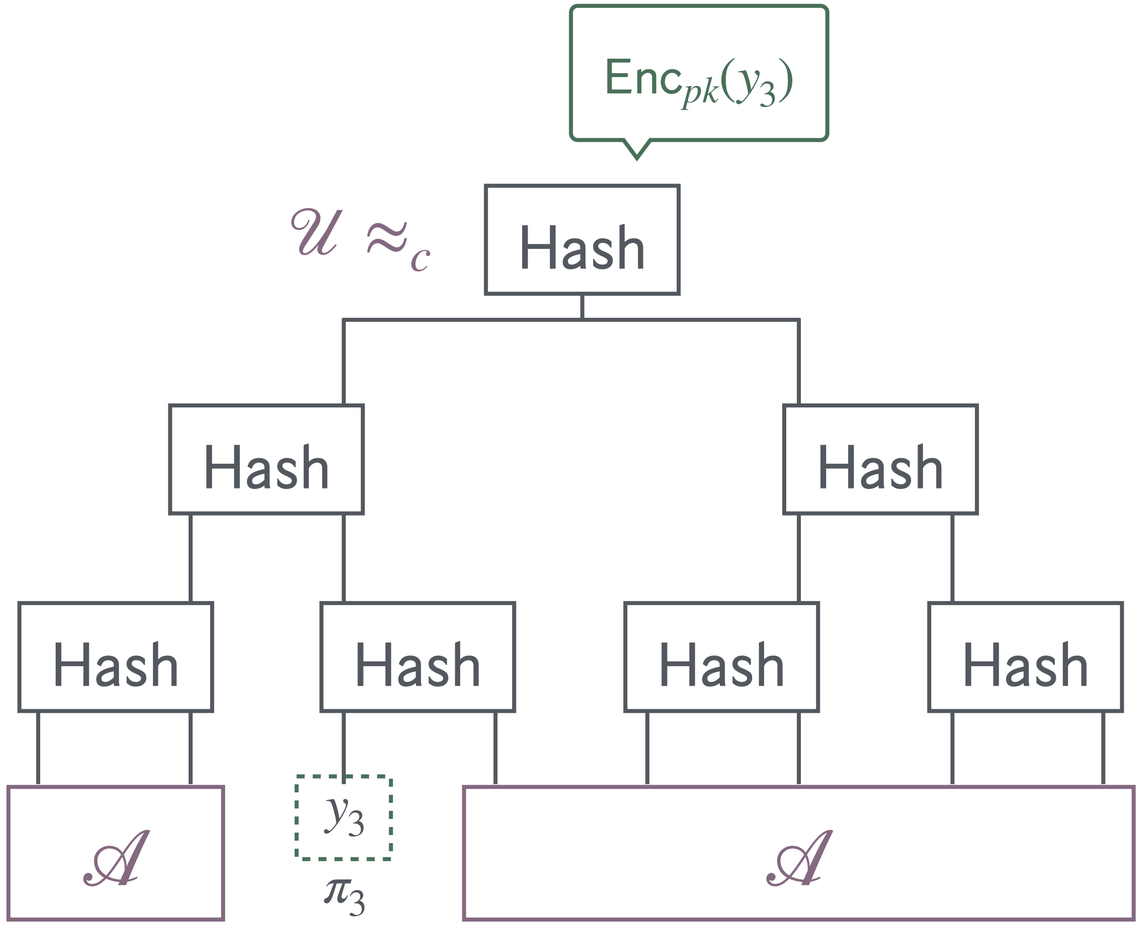


(AGGREGATE) PSEUDORANDOMNESS

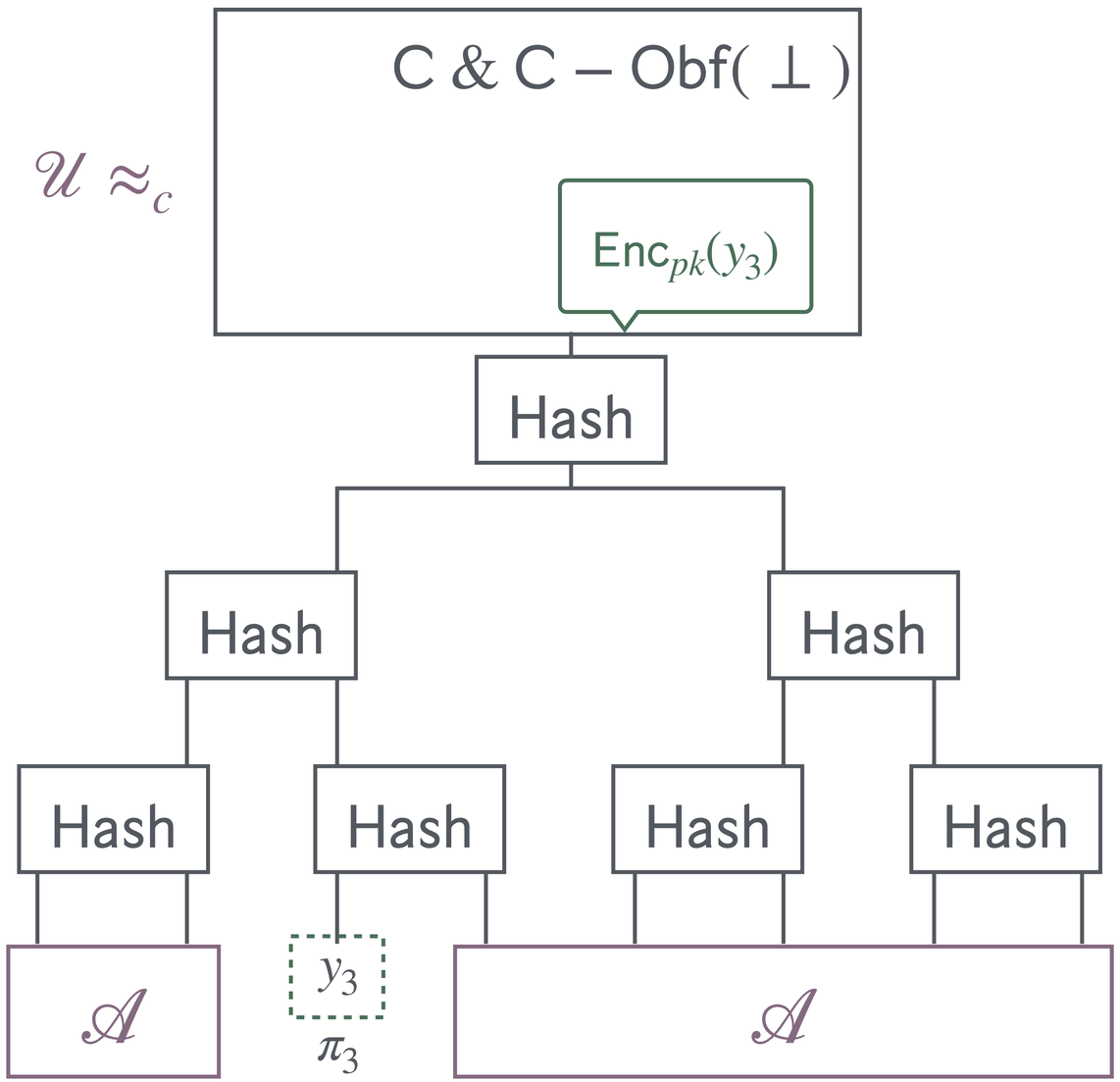
- Randomness extractor does not work



(AGGREGATE) PSEUDORANDOMNESS

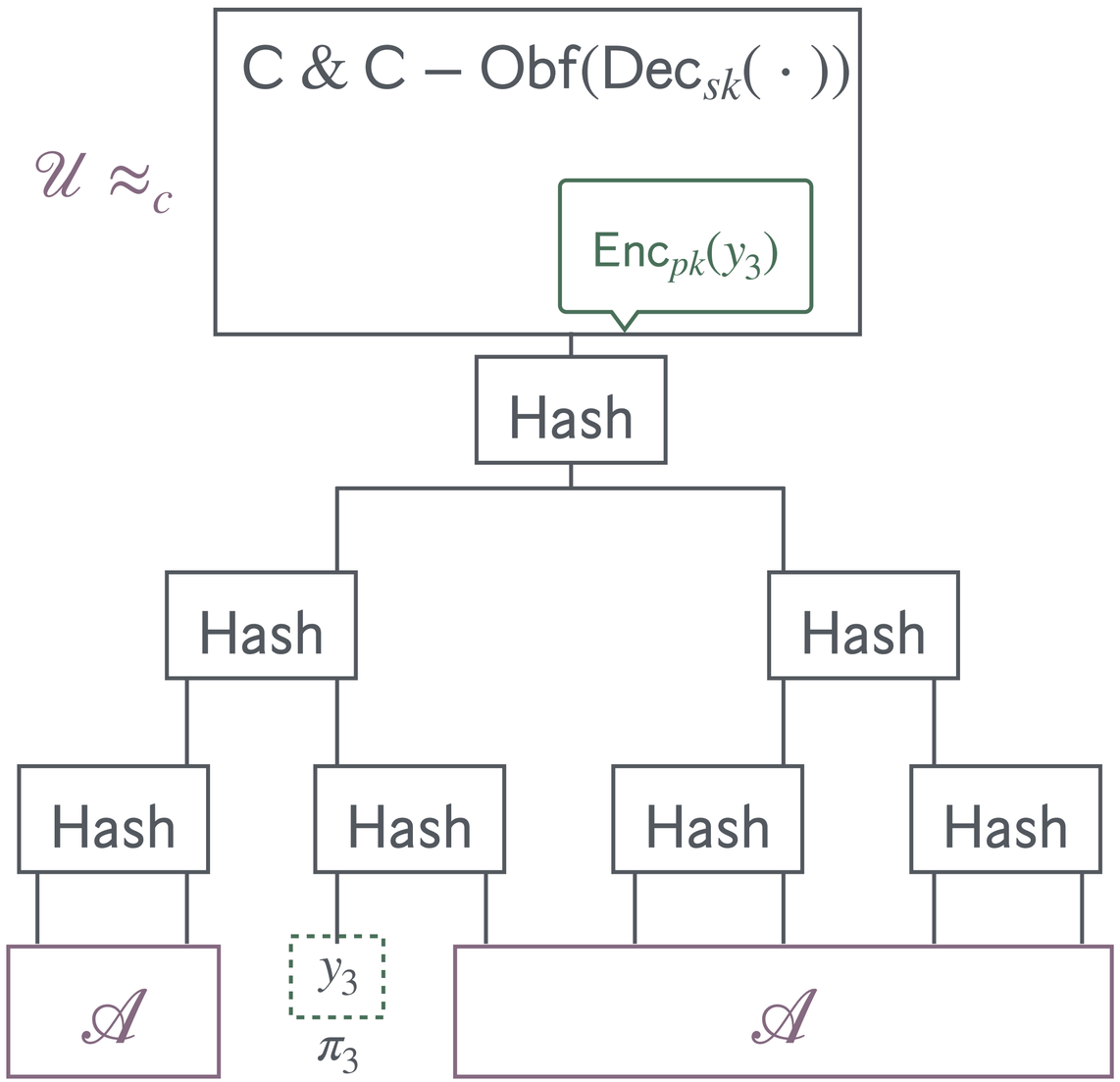


- Randomness extractor does not work
- Idea II: Compute-and-compare obfuscation [GKW17,WZ17]



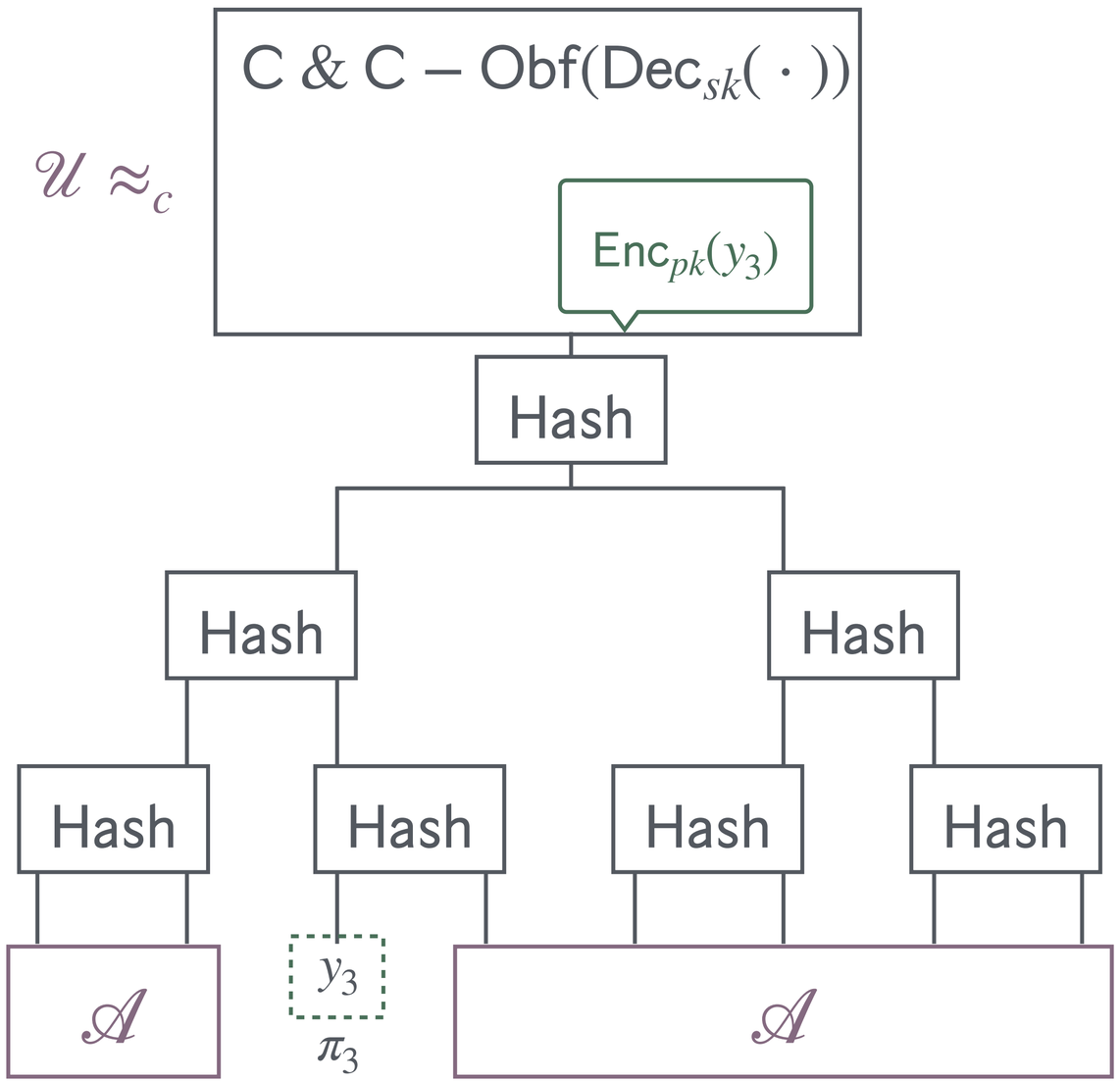
(AGGREGATE) PSEUDORANDOMNESS

- Randomness extractor does not work
- Idea II: Compute-and-compare obfuscation [GKW17,WZ17]
  - Run the root through C&C-obfuscation (of a dummy program)



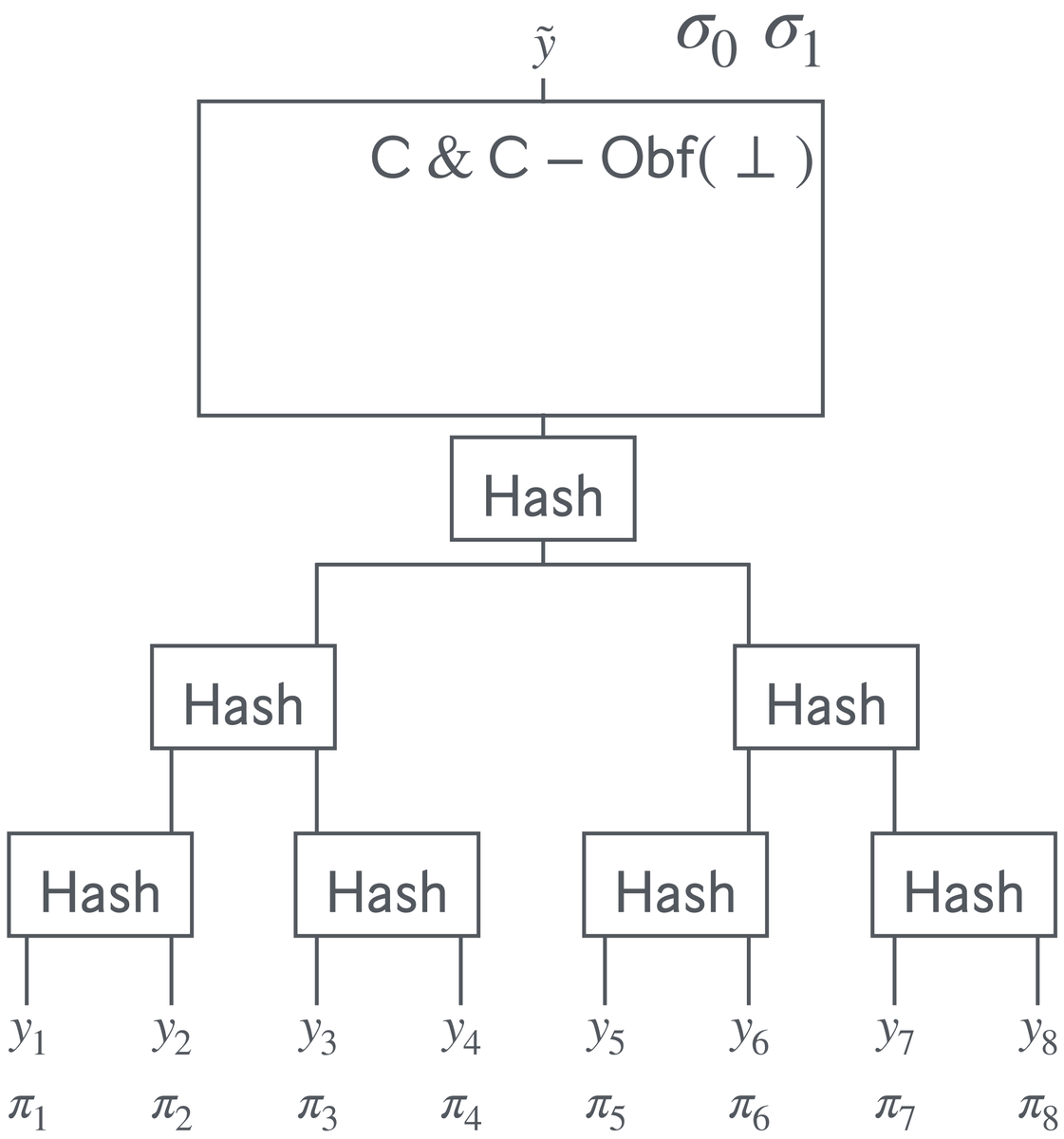
(AGGREGATE) PSEUDORANDOMNESS

- Randomness extractor does not work
- Idea II: Compute-and-compare obfuscation [GKW17,WZ17]
  - Run the root through C&C-obfuscation (of a dummy program)
  - Switch to a decryption circuit

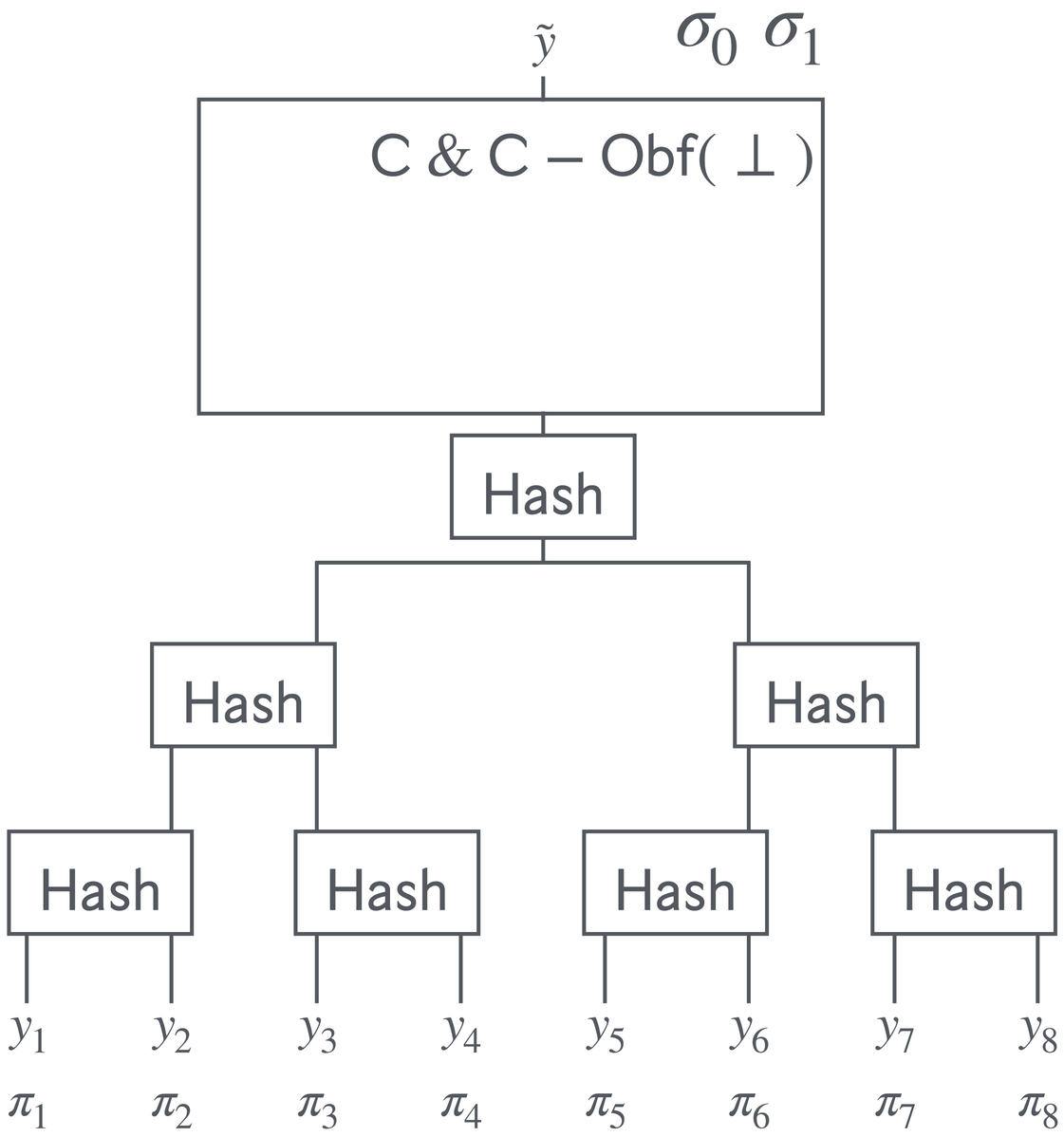


(AGGREGATE) PSEUDORANDOMNESS

- Randomness extractor does not work
- Idea II: Compute-and-compare obfuscation [GKW17,WZ17]
  - Run the root through C&C-obfuscation (of a dummy program)
  - Switch to a decryption circuit
  - Use the randomness extraction property of random matrices

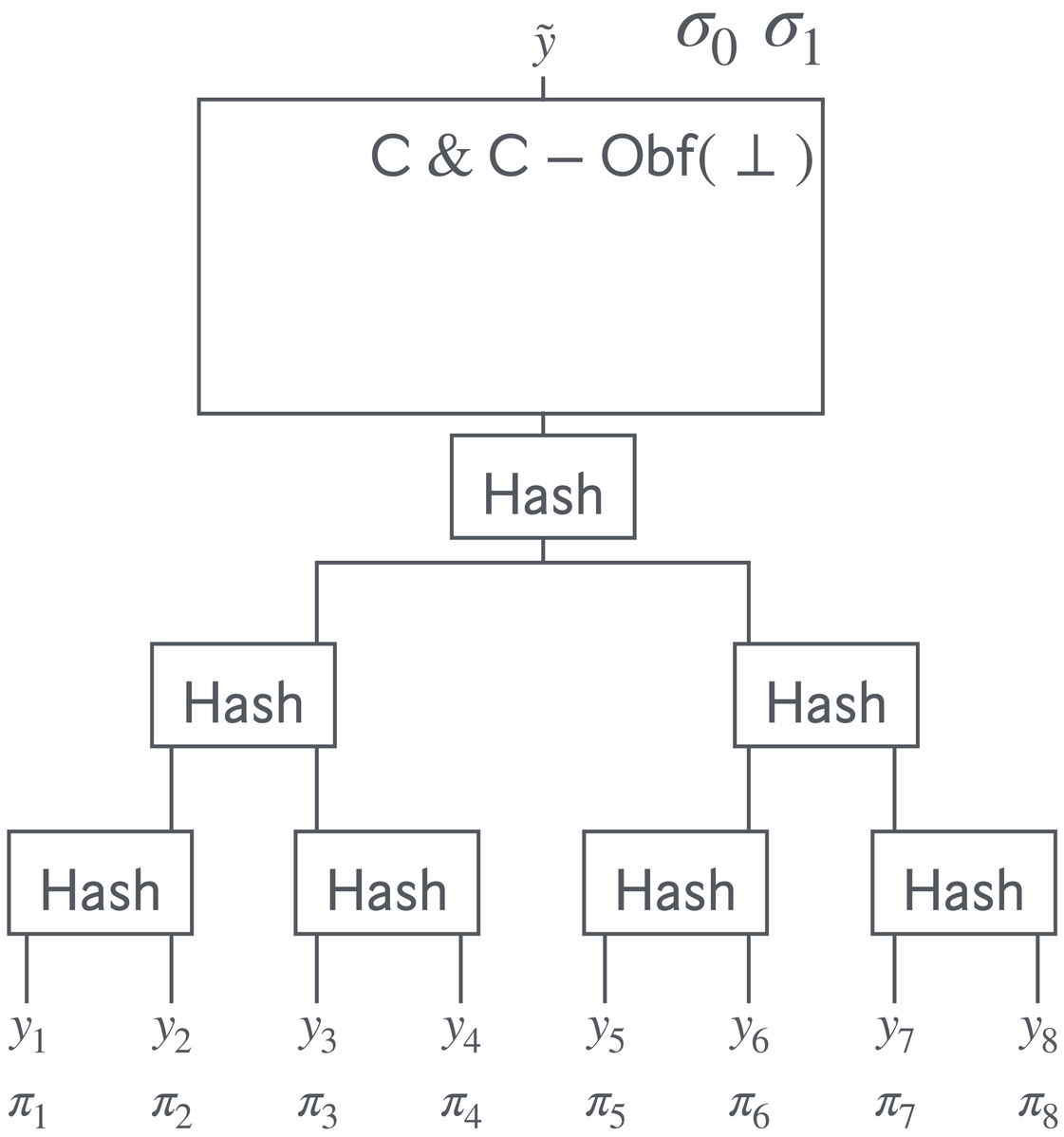


(AGGREGATE) UNIQUENESS



(AGGREGATE) UNIQUENESS

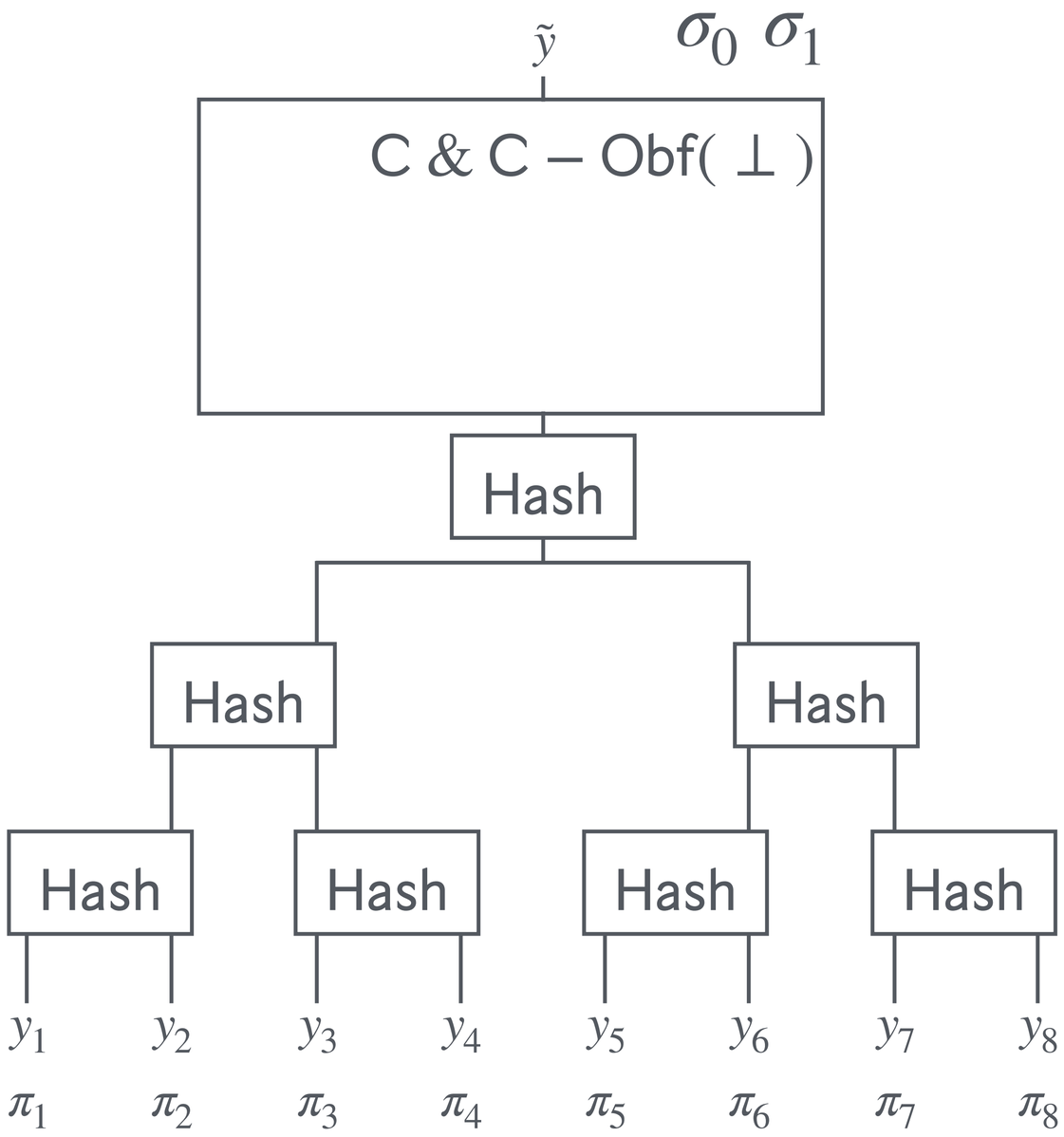
- Want: No adversary can find $(y^*, \sigma_0^*, \sigma_1^*)$ s.t.:



(AGGREGATE) UNIQUENESS

- Want: No adversary can find $(y^*, \sigma_0^*, \sigma_1^*)$ s.t.:

  1. $(y^*, \sigma_0^*, \sigma_1^*)$ is a valid tuple

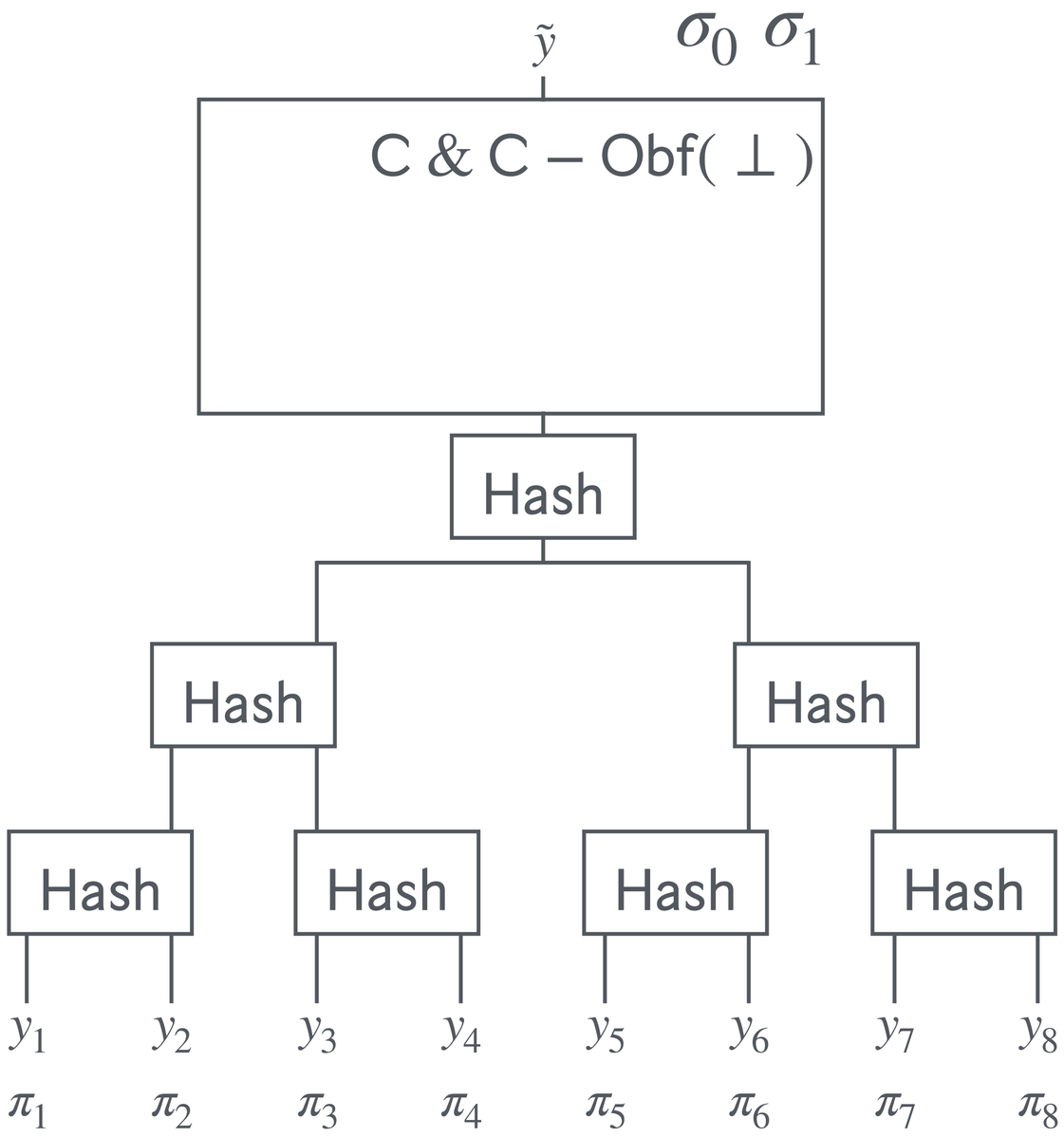


(AGGREGATE) UNIQUENESS

- Want: No adversary can find $(y^*, \sigma_0^*, \sigma_1^*)$ s.t.:

  1. $(y^*, \sigma_0^*, \sigma_1^*)$ is a valid tuple

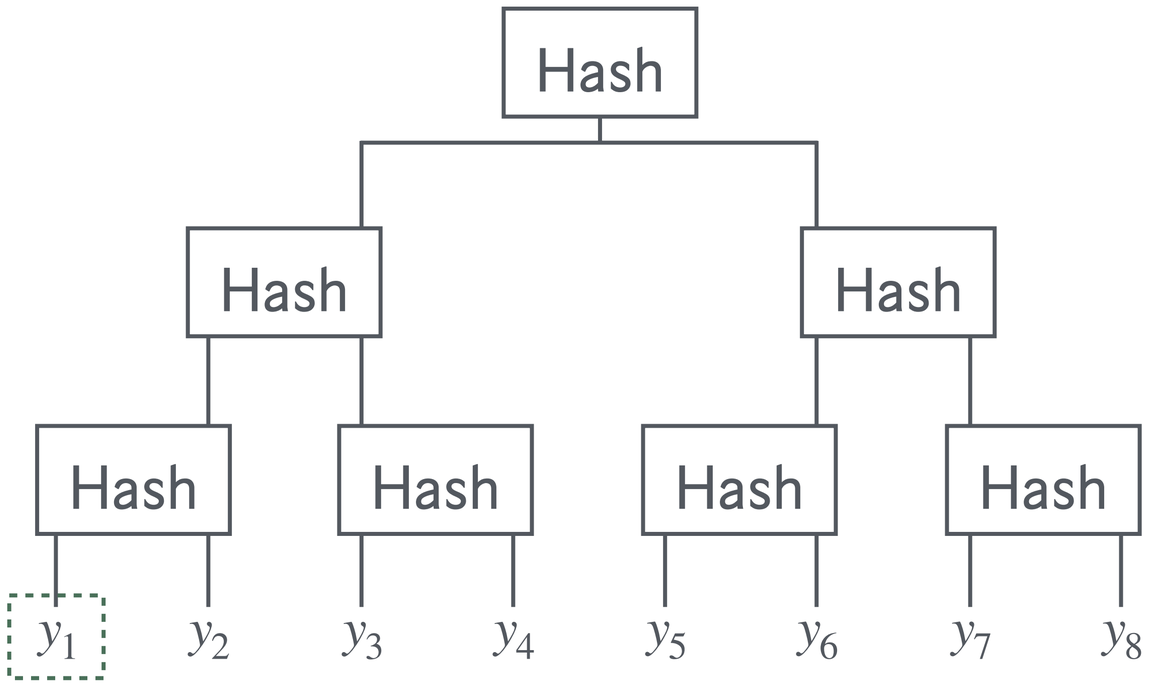
  2. $y^* \neq \tilde{y}$



(AGGREGATE) UNIQUENESS

- Want: No adversary can find $(y^*, \sigma_0^*, \sigma_1^*)$ s.t.:
  1. $(y^*, \sigma_0^*, \sigma_1^*)$ is a valid tuple
  2. $y^* \neq \tilde{y}$
- We want to derive a contradiction against the uniqueness of the (non-aggregate) VRF

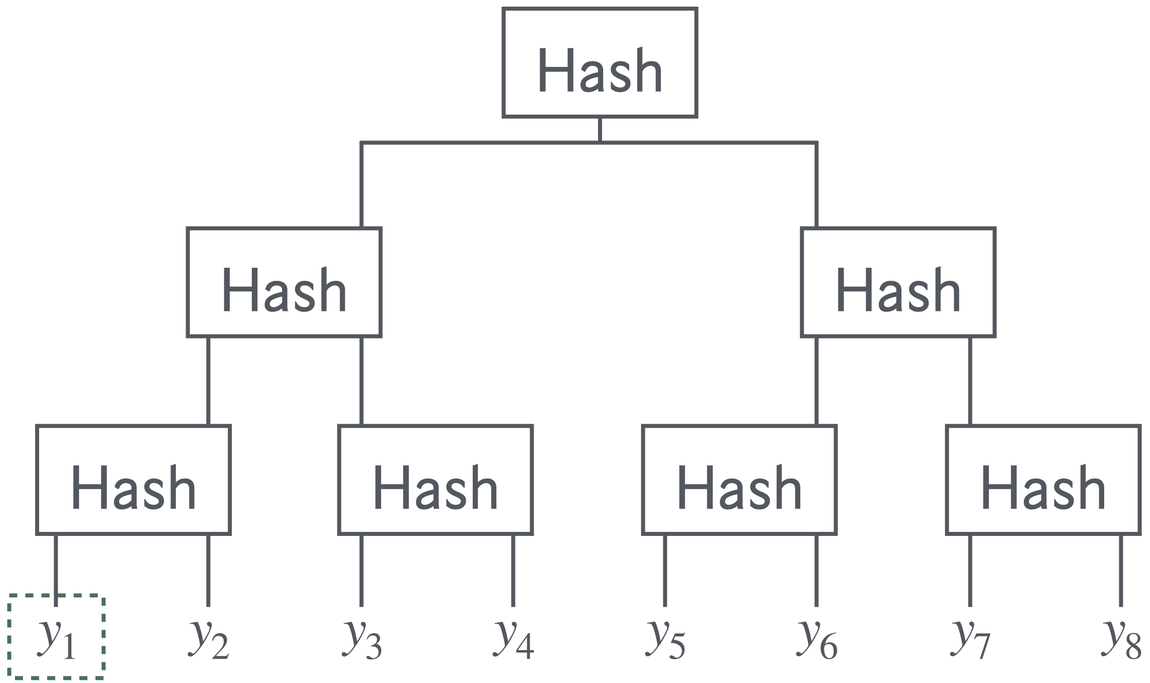
HONEST TREE



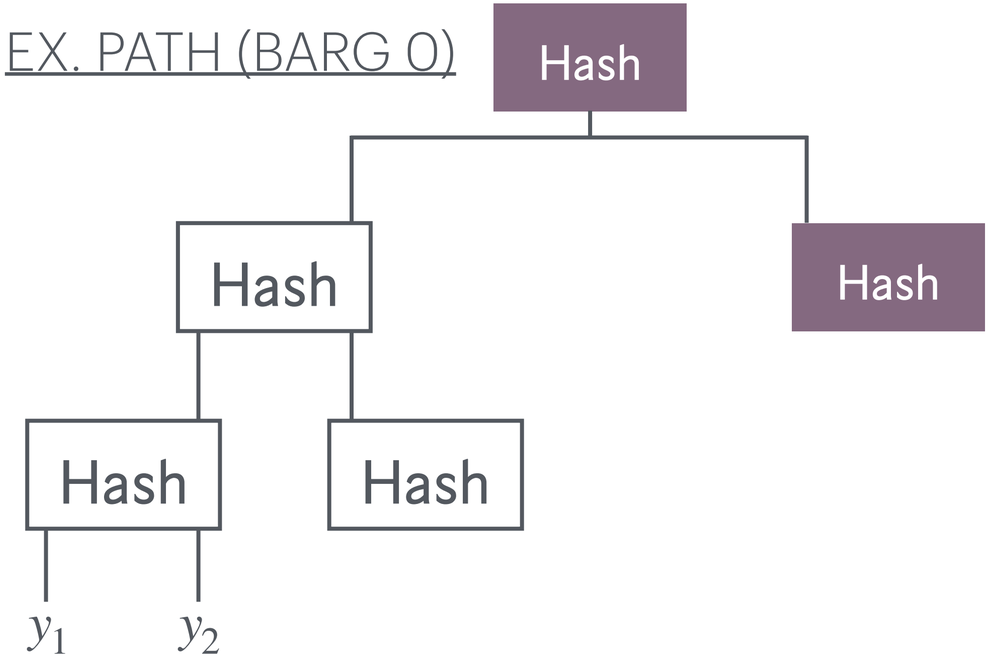
EX. PATH (BARG 0)

EX. PATH (BARG 1)

HONEST TREE

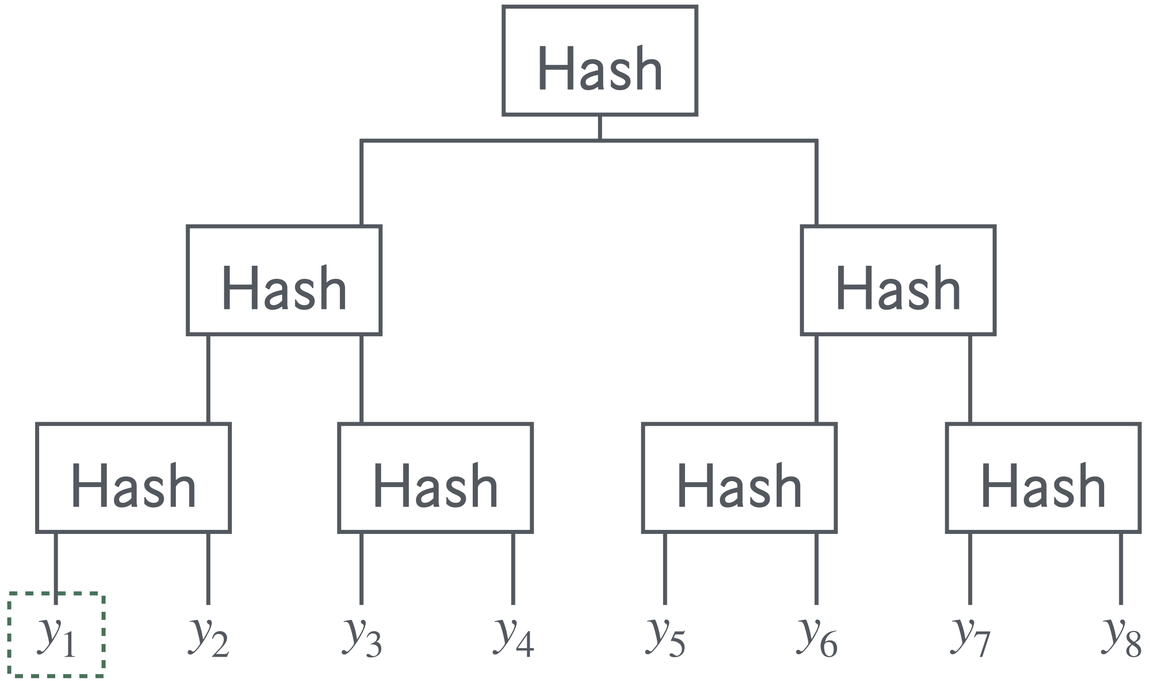


EX. PATH (BARG 0)

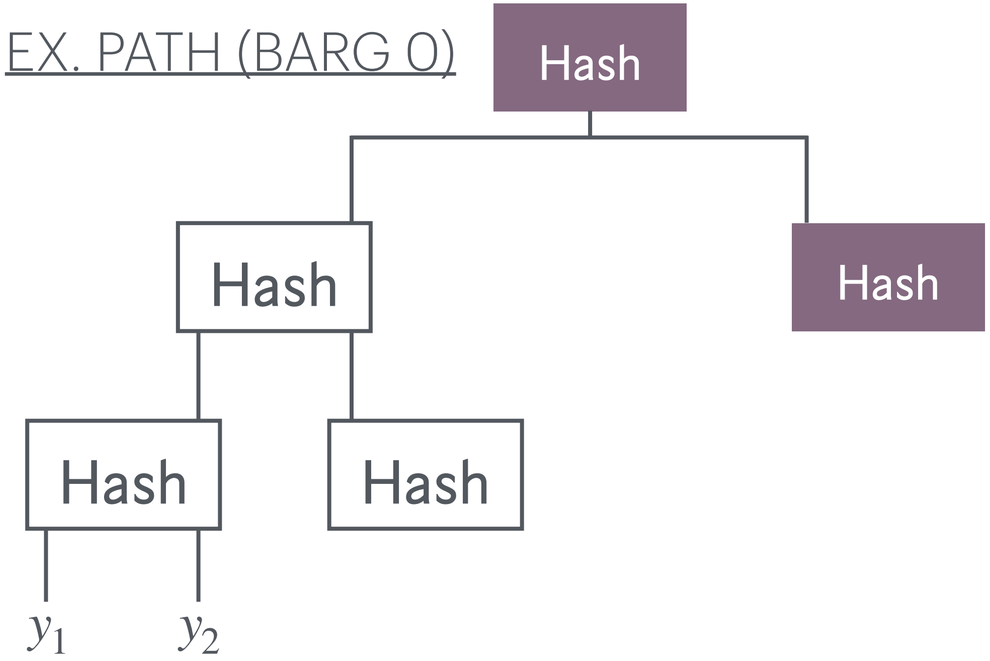


EX. PATH (BARG 1)

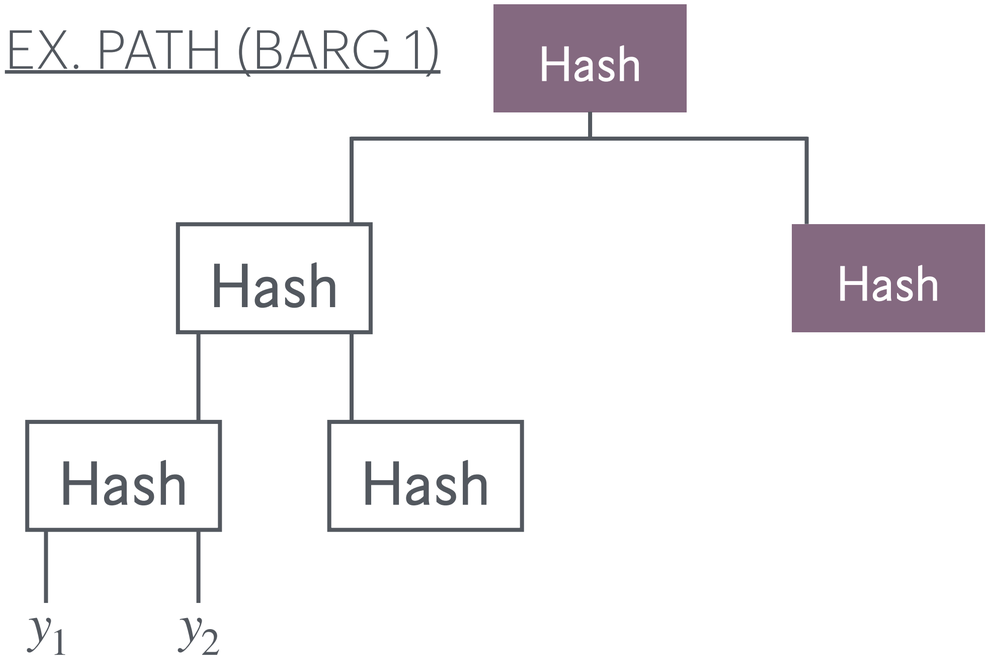
HONEST TREE



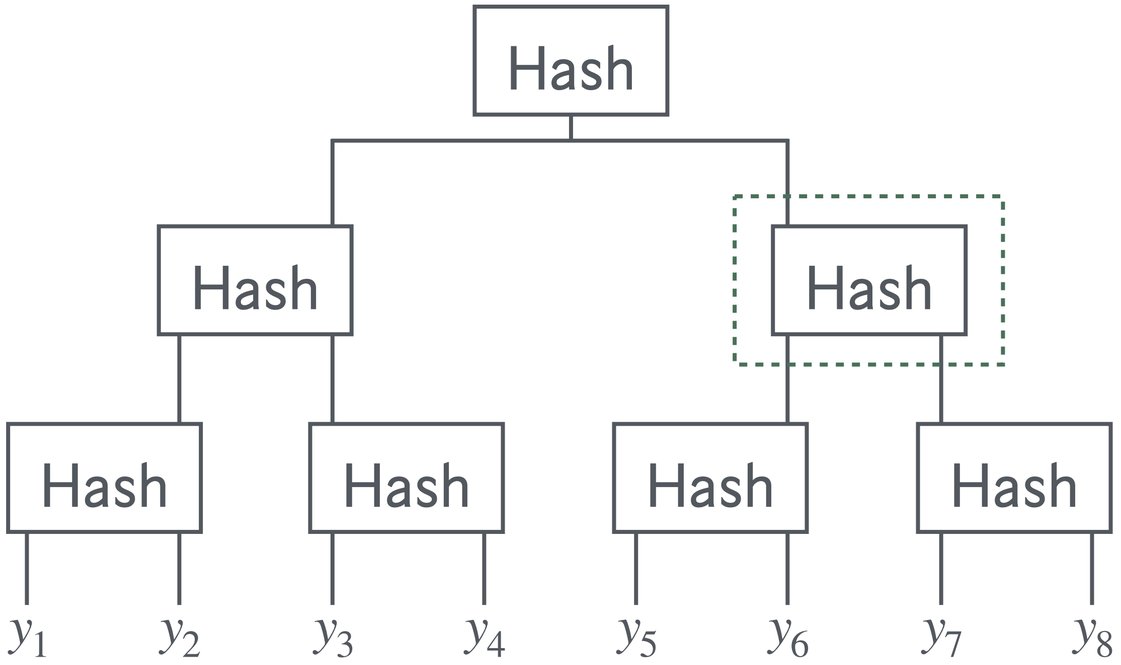
EX. PATH (BARG 0)



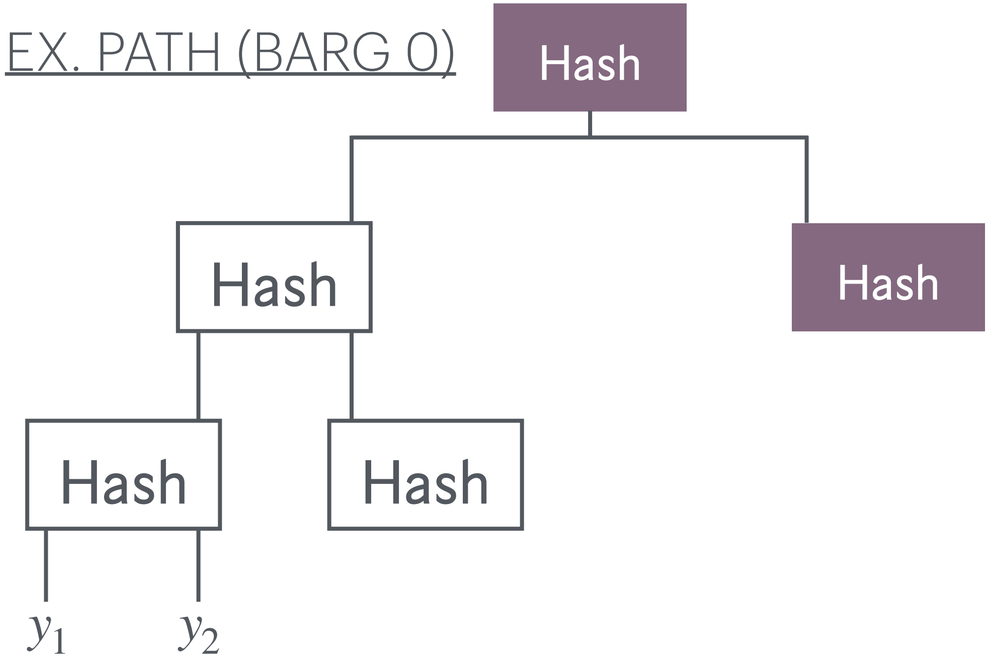
EX. PATH (BARG 1)



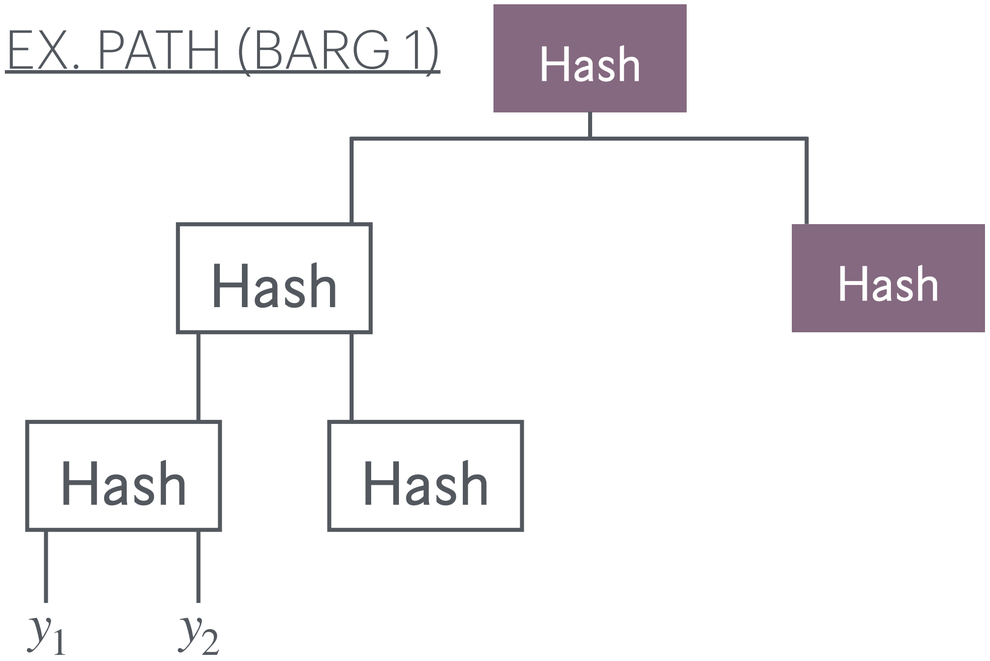
HONEST TREE



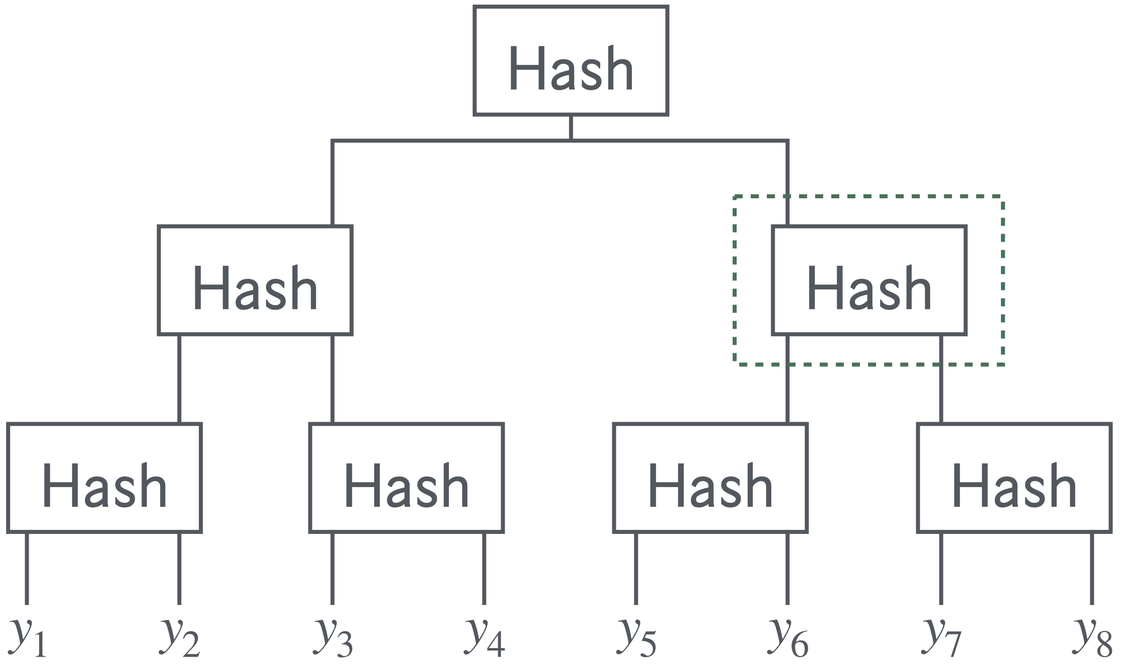
EX. PATH (BARG 0)



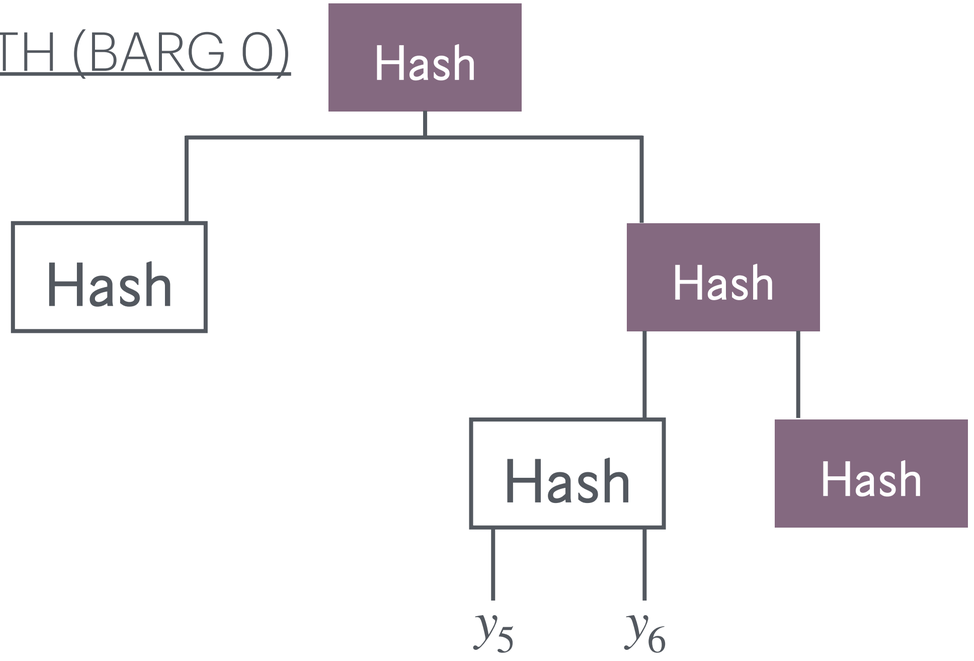
EX. PATH (BARG 1)



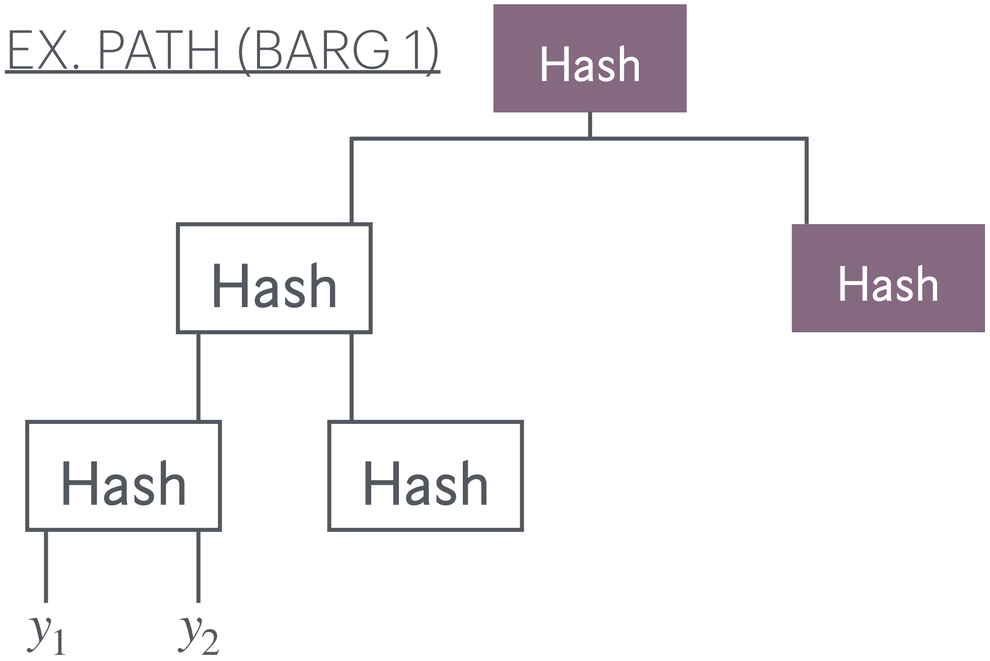
HONEST TREE



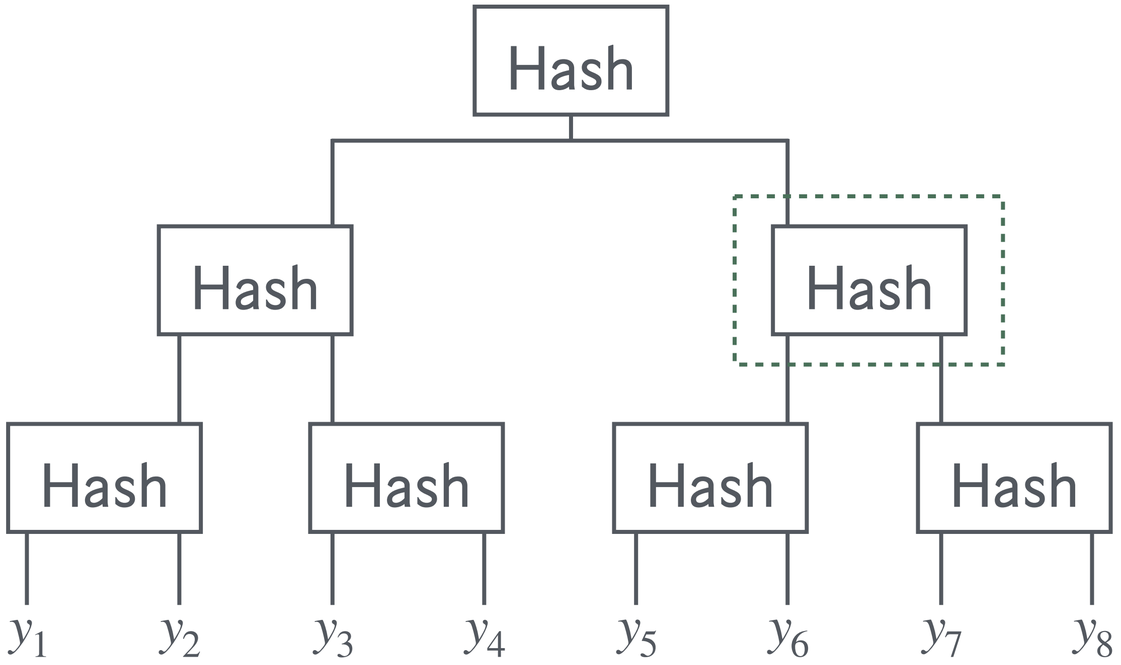
EX. PATH (BARG 0)



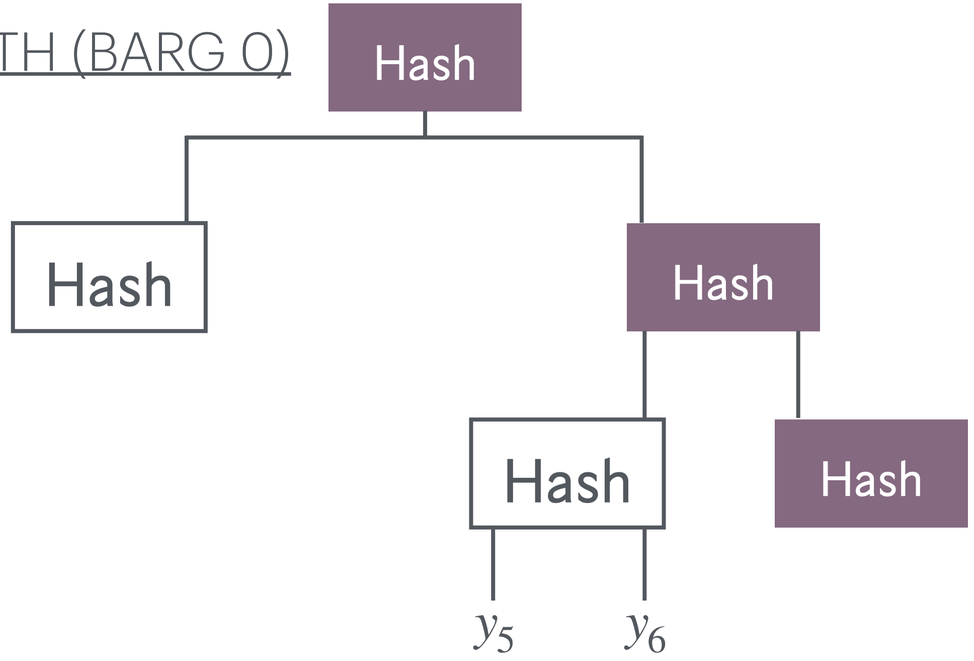
EX. PATH (BARG 1)



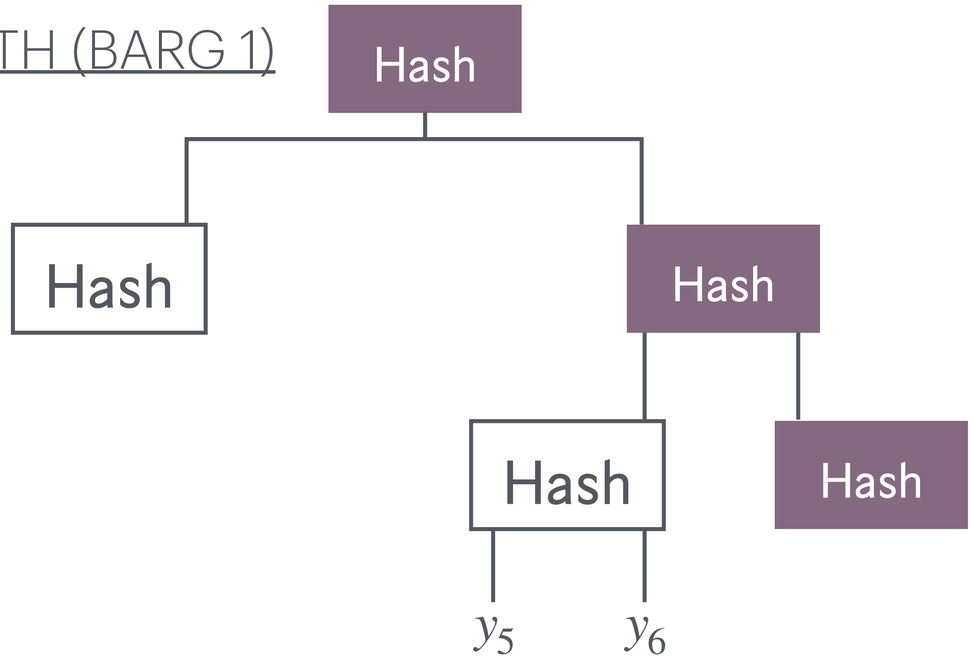
HONEST TREE



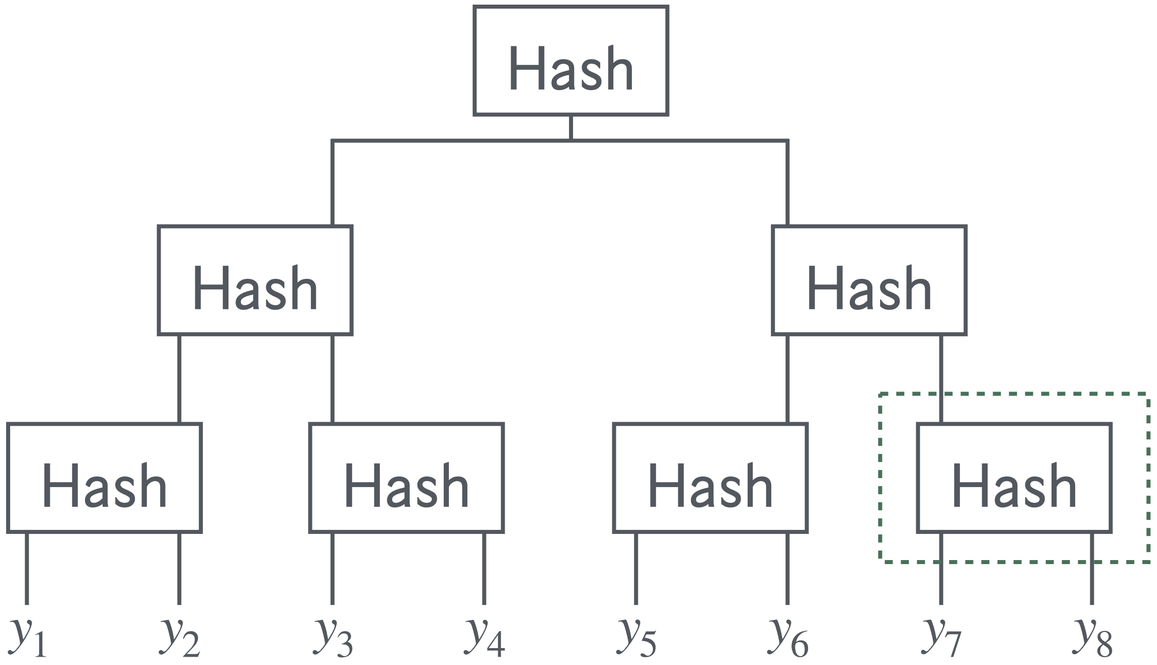
EX. PATH (BARG 0)



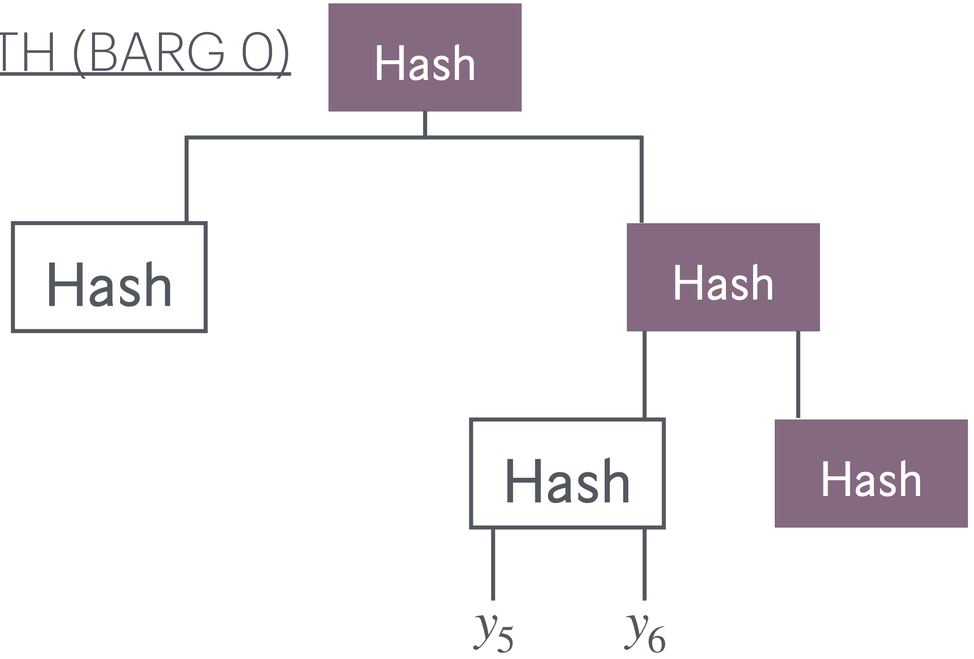
EX. PATH (BARG 1)



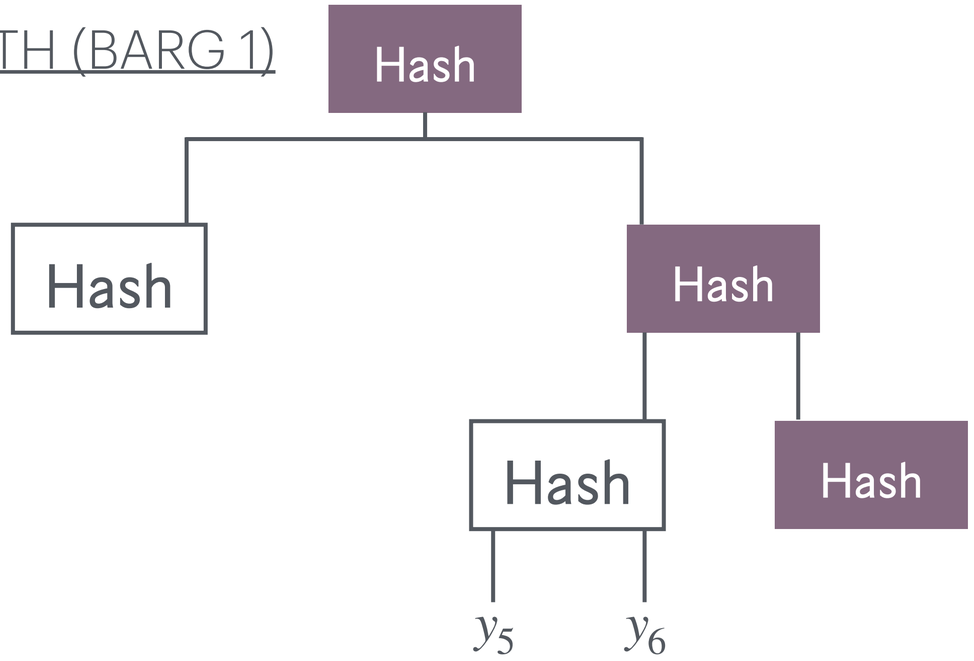
HONEST TREE



EX. PATH (BARG 0)



EX. PATH (BARG 1)



Bonus

$$vk = (h, h^k)$$
$$sk = k$$

$$y = e(g, \mathscr{H}(x))^k$$

$$y =^? e(g, \underbrace{\mathscr{H}(x)^k}_{=:\pi}) \quad \& \quad e(h^k, \mathscr{H}(x)) =^? e(h, \pi)$$

Open Problems

Open Problems

- Security with a malicious aggregator?

Open Problems

- Security with a malicious aggregator?
- More applications?

Open Problems

- Security with a malicious aggregator?
- More applications?
- Different computational assumptions?

Open Problems

- Security with a malicious aggregator?
- More applications?
- Different computational assumptions?
- Better concrete efficiency, in the standard model?

Open Problems

- Security with a malicious aggregator?
- More applications?
- Different computational assumptions?
- Better concrete efficiency, in the standard model?

Thank you!