Fully Succinct and Updatable Batch Arguments for NP from Indistinguishability Obfuscation

Rachit Garg
UT Austin

Kristin Sheridan
UT Austin

Brent Waters
UT Austin
NTT Research

David Wu
UT Austin

TCC November 2022

Fully Succinct and Updatable Batch Arguments for NP from Indistinguishability Obfuscation

Rachit Garg
UT Austin

Kristin Sheridan
UT Austin

Brent Waters
UT Austin
NTT Research

David Wu
UT Austin

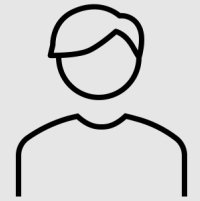
TCC November 2022

Batch Arguments for NP

Batch Arguments for NP

PROVER

VERIFIER

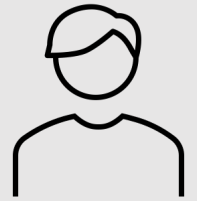


Batch Arguments for NP

PROVER

$(C, x_1, \ldots, x_t)$

VERIFIER



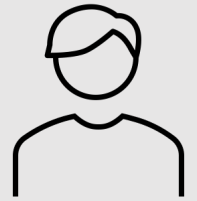
Batch Arguments for NP

$$\mathcal{L}_C = \{x \in \{0,1\}^n : \text{ exists } w \in \{0,1\}^m, C(x,w) = 1\}$$

PROVER

$$(C, x_1, \ldots, x_t)$$

VERIFIER



Batch Arguments for NP

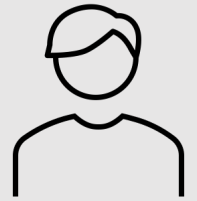
$$\mathcal{L}_C = \{x \in \{0,1\}^n : \text{ exists } w \in \{0,1\}^m, C(x,w) = 1\}$$

PROVER

$(C, x_1, \ldots, x_t)$

$(w_1, \ldots, w_t)$

VERIFIER



Batch Arguments for NP

$$\mathcal{L}_C = \{x \in \{0,1\}^n : \text{ exists } w \in \{0,1\}^m, C(x,w) = 1\}$$

PROVER

$(C, x_1, \ldots, x_t)$

$(w_1, \ldots, w_t)$

VERIFIER

Batch Arguments for NP

$$\mathcal{L}_C = \{x \in \{0,1\}^n : \text{ exists } w \in \{0,1\}^m, C(x,w) = 1\}$$

PROVER

$(C, x_1, \ldots, x_t)$

π

VERIFIER

Batch Arguments for NP

$$\mathcal{L}_C = \{x \in \{0,1\}^n : \text{ exists } w \in \{0,1\}^m, C(x,w) = 1\}$$

PROVER

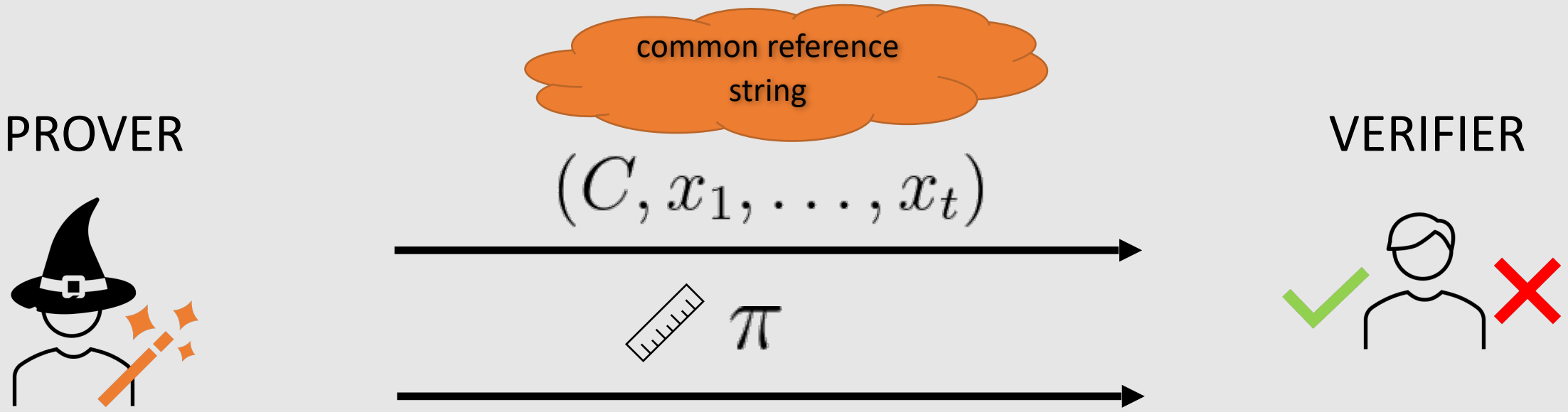
$(C, x_1, \ldots, x_t)$

π

VERIFIER

Batch Arguments for NP

$$\mathcal{L}_C = \{x \in \{0,1\}^n : \text{ exists } w \in \{0,1\}^m, C(x,w) = 1\}$$



Batch Arguments for NP - BARGs

Proof π

Batch Arguments for NP - BARGs

of instances

Circuit size

Proof π

Batch Arguments for NP - BARGs

of instances

Circuit size

Proof π

$o(t)$

$\mathsf{poly}(\log t)$

Batch Arguments for NP - BARGs

Proof π

# of instances

$o(t)$ $\quad$ $\mathsf{poly}(\log t)$

Circuit size

$\mathsf{poly}(|C|)$ $\quad$ $\mathsf{poly}(\log |C|)$

Batch Arguments for NP - BARGs

Proof π

# of instances

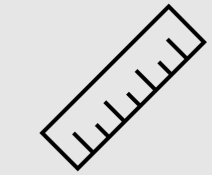
$o(t)$ $\quad$ $\mathsf{poly}(\log t)$

Circuit size

$\mathsf{poly}(|C|)$ $\quad$ $\mathsf{poly}(\log |C|)$

fully succinct $\quad$ $\mathsf{poly}(\lambda, \log t, \log |C|)$

Batch Arguments for NP - BARGs



Proof π

# of instances

$o(t)$ $\quad$ $\mathsf{poly}(\log t)$

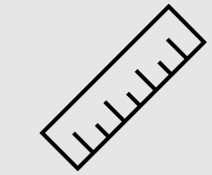
Circuit size

$\mathsf{poly}(|C|)$ $\quad$ $\mathsf{poly}(\log |C|)$

fully succinct $\quad$ $\boxed{\mathsf{poly}(\lambda, \log t, \log |C|)}$

Completeness: *"Honest prover convinces verifier of true statements"*

Batch Arguments for NP - BARGs



Proof π

# of instances

$o(t)$ $\quad$ $\mathsf{poly}(\log t)$

Circuit size

$\mathsf{poly}(|C|)$ $\quad$ $\mathsf{poly}(\log |C|)$

fully succinct $\quad$ $\boxed{\mathsf{poly}(\lambda, \log t, \log |C|)}$

Completeness: *"Honest prover convinces verifier of true statements"*

Soundness: *"No efficient prover can convince honest verifier of false statements"*

Prior Work (BARGs for NP)

Prior Work (BARGs for NP)

- Use SNARGs for NP - Proof independent of witness length

Prior Work (BARGs for NP)

- Use SNARGs for NP - Proof independent of witness length
  - Existing constructions - Idealized Models, strong knowledge assumptions

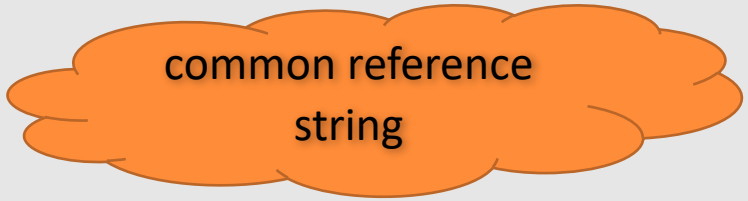
Prior Work (BARGs for NP)

- Use SNARGs for NP - Proof independent of witness length
  - Existing constructions - Idealized Models, strong knowledge assumptions
  - Sahai-Waters14 – from Indistinguishability Obfuscation (from iO + owf)

Prior Work (BARGs for NP)

- Use SNARGs for NP - Proof independent of witness length
  - Existing constructions - Idealized Models, strong knowledge assumptions
  - Sahai-Waters14 – from Indistinguishability Obfuscation (from iO + owf)

Sahai-Waters14



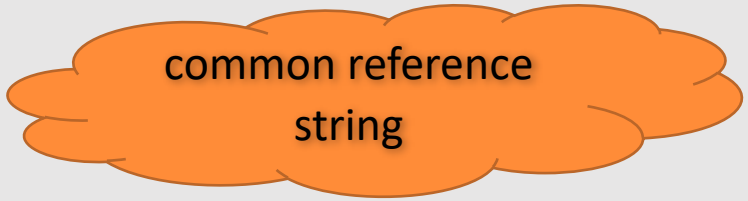
crs grows with t

supports apriori fixed instances

Prior Work (BARGs for NP)

- Use SNARGs for NP - Proof independent of witness length
  - Existing constructions - Idealized Models, strong knowledge assumptions
  - Sahai-Waters14 – from Indistinguishability Obfuscation (from iO + owf)

Sahai-Waters14



crs grows with t

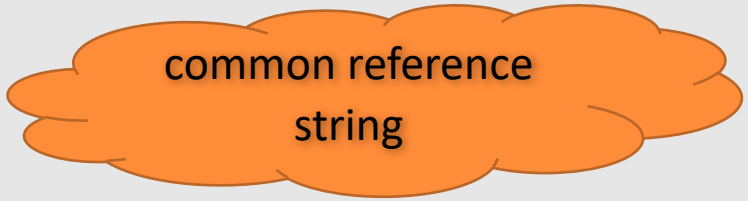
supports apriori fixed instances

Want to support arbitrary polynomial instances

Prior Work (BARGs for NP)

- Use SNARGs for NP - Proof independent of witness length
  - Existing constructions - Idealized Models, strong knowledge assumptions
  - Sahai-Waters14 – from Indistinguishability Obfuscation (from iO + owf)

Sahai-Waters14



crs grows with t

supports apriori fixed instances

Want to support arbitrary polynomial instances

- CJJ21b,DGKV22 – from LWE, WW22 – from bilinear maps, CJJ21a + KLVW22 – DDH + QR

Not fully succinct

Our Results

BARGs for NP

Our Results

BARGs for NP

Support arbitrary polynomial instances

Our Results

BARGs for NP

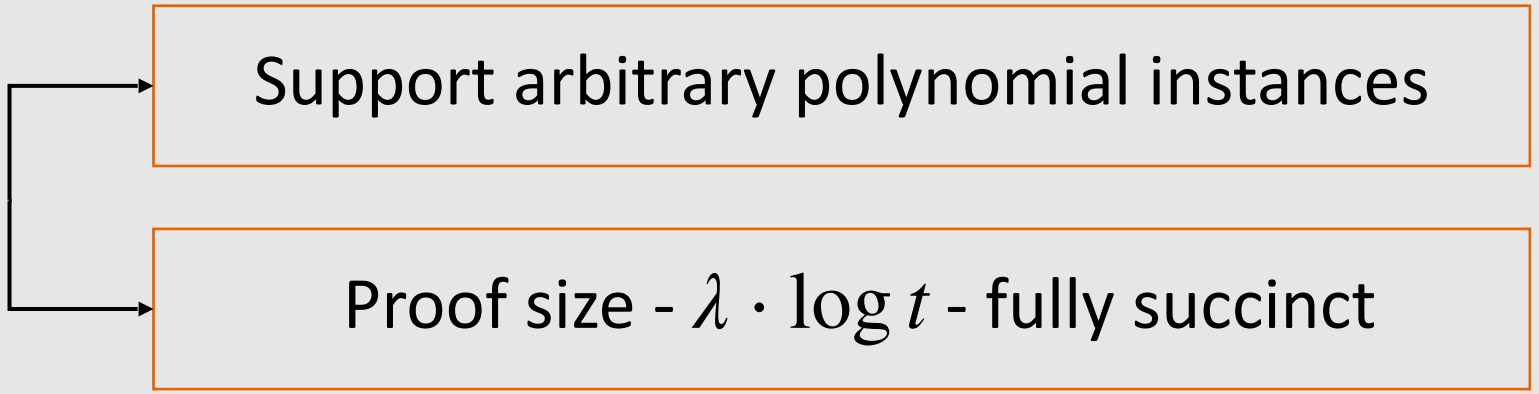
Support arbitrary polynomial instances

Proof size - $\lambda \cdot \log t$ - fully succinct

Our Results

BARGs for NP

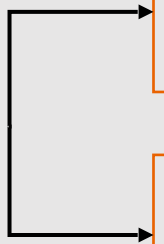
SW14+CJJ21b



Our Results

BARGs for NP

SW14+CJJ21b



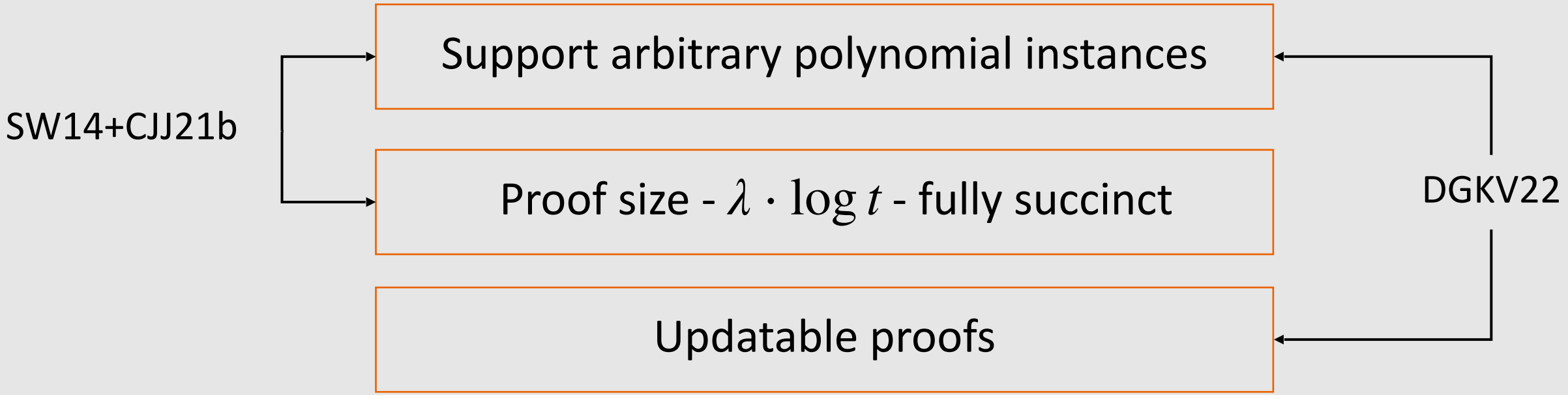
Support arbitrary polynomial instances

Proof size - $\lambda \cdot \log t$ - fully succinct

Updatable proofs

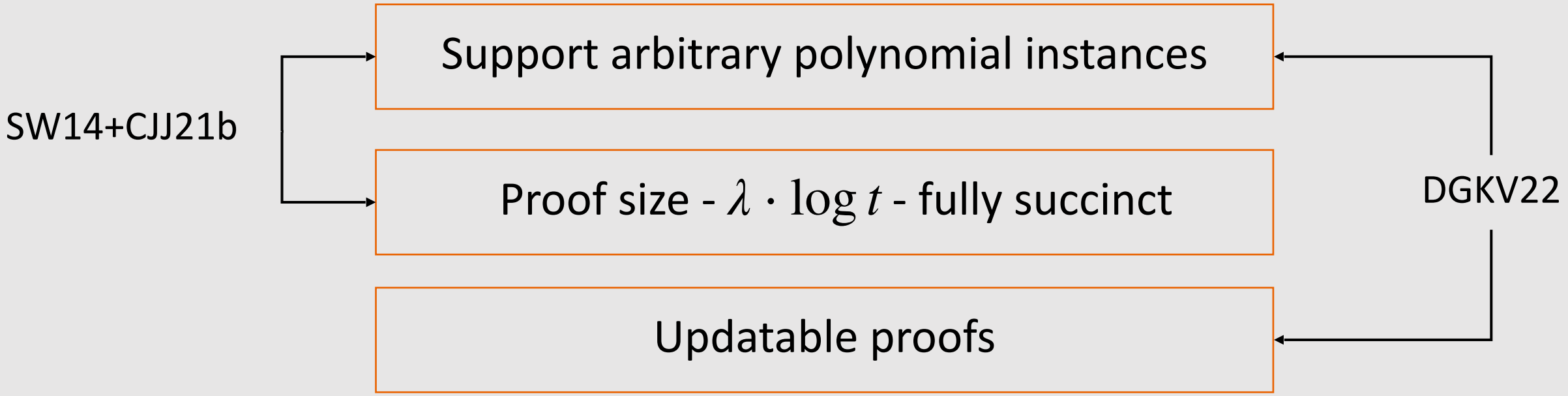
Our Results

BARGs for NP



Our Results

BARGs for NP



A more direct construction (similar to SW14) from iO+owf without Fiat-Shamir and correlation intractability

Updatable BARGs for NP

Updatable BARGs for NP

PROVER

common reference string

Updatable BARGs for NP

PROVER

common reference string

$(x_1, \ldots, x_t)$

C

π_1

Updatable BARGs for NP

PROVER

common reference string

C

$(x_1, \ldots, x_t)$ $(x_{t+1}, \ldots, x_{2t})$ poly $\cdots$ $(x_{(t'-1)\cdot t+1}, \ldots, x_{t'\cdot t})$

π_1 π_2 $\pi_{t'}$

Updatable BARGs for NP

PROVER

common reference string

poly

C

$(x_1, \ldots, x_t)$ $(x_{t+1}, \ldots, x_{2t})$ $\cdots$ $(x_{(t'-1)\cdot t+1}, \ldots, x_{t'\cdot t})$

π_1 π_2 $\pi_{t'}$

π

Updatable BARGs for NP

PROVER

common reference string

C

$(x_1, \ldots, x_t)$ $(x_{t+1}, \ldots, x_{2t})$ poly $\cdots$ $(x_{(t'-1)\cdot t+1}, \ldots, x_{t'\cdot t})$

π_1 π_2 $\pi_{t'}$

π

$\mathsf{poly}(\lambda, \log(t \cdot t'), \log|C|)$

Updatable BARGs for NP

Support arbitrary polynomial updates

PROVER

common reference string

C

$(x_1, \ldots, x_t)$ $(x_{t+1}, \ldots, x_{2t})$ poly $\cdots$ $(x_{(t'-1)\cdot t+1}, \ldots, x_{t'\cdot t})$

π_1 π_2 $\pi_{t'}$

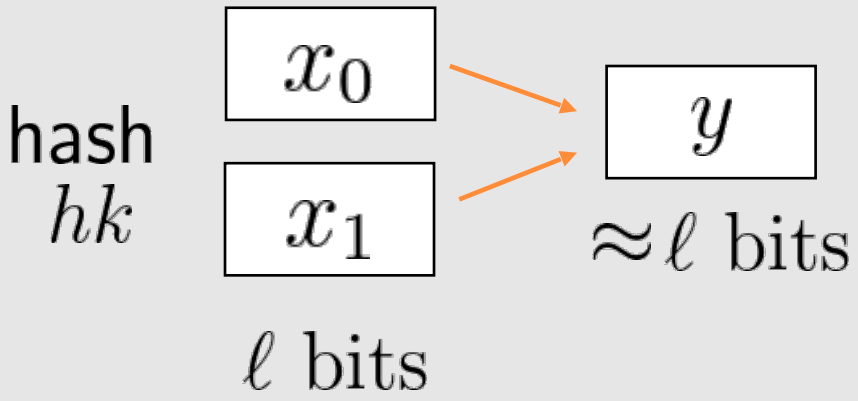
$\mathsf{poly}(\lambda, \log(t \cdot t'), \log|C|)$

π

Ingredients

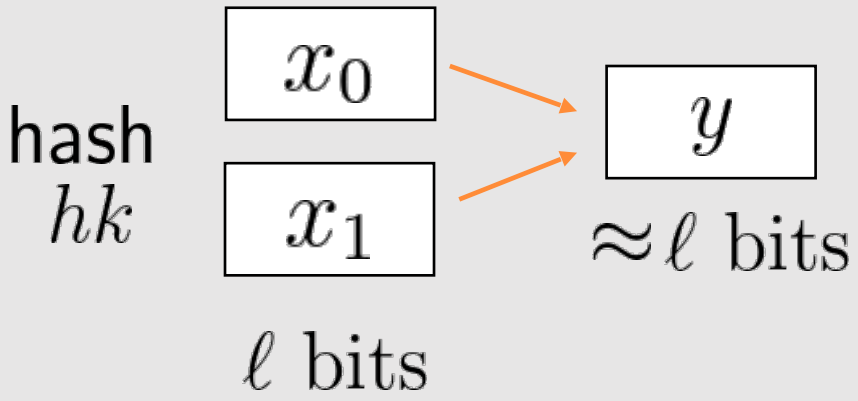
Ingredients

Two-to-one Hash function ★



Ingredients

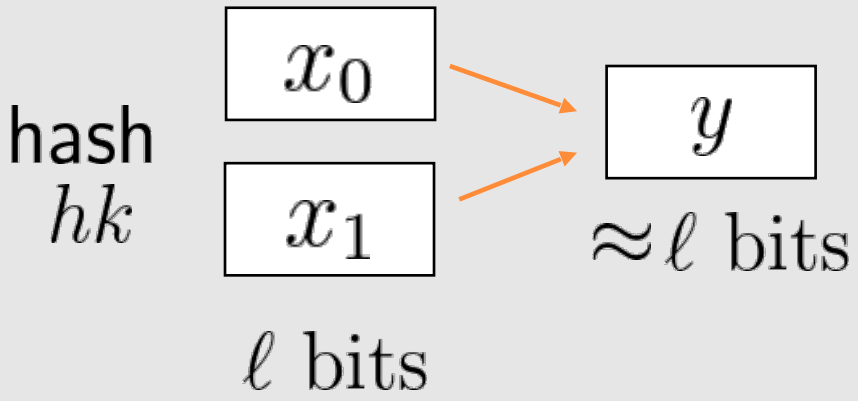
★ **Two-to-one Hash function**



★ **Statistical Binding Property (positional accumulator)**

Ingredients

Two-to-one Hash function ★

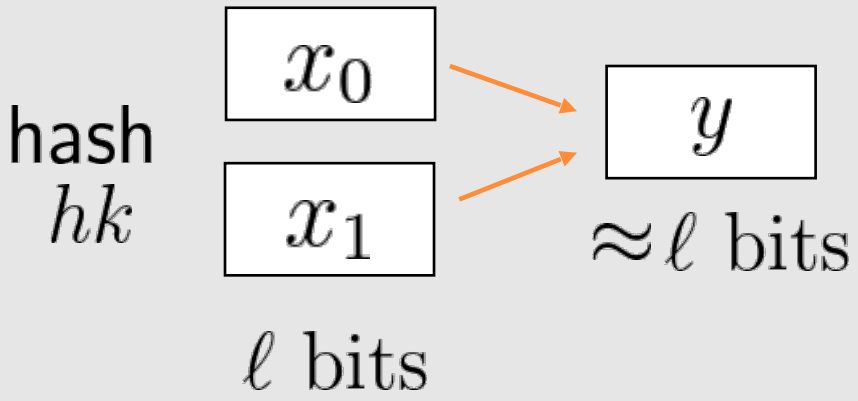


★ **Statistical Binding Property (positional accumulator)**

Indistinguishability Obfuscation + **One-Way functions**

Ingredients

Two-to-one Hash function ★



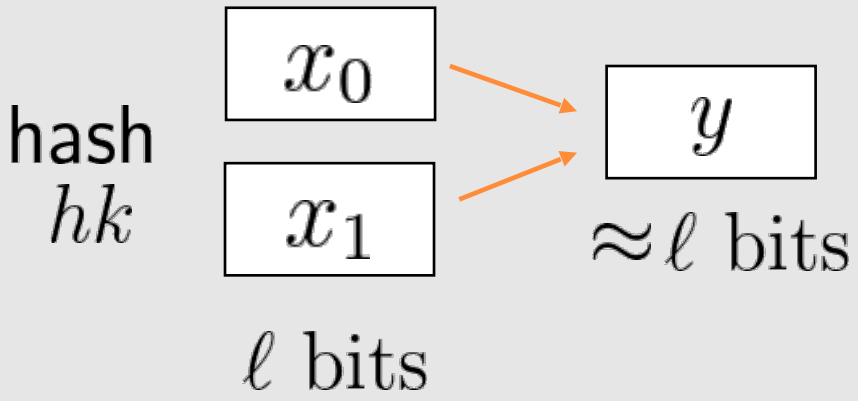
★ **Statistical Binding Property (positional accumulator)**

Indistinguishability Obfuscation + **One-Way functions**

SNARG for NP [SW14]

Ingredients

Two-to-one Hash function ★



★ **Statistical Binding Property (positional accumulator)**

Indistinguishability Obfuscation + **One-Way functions**

SNARG for NP [SW14]

crs

Prove

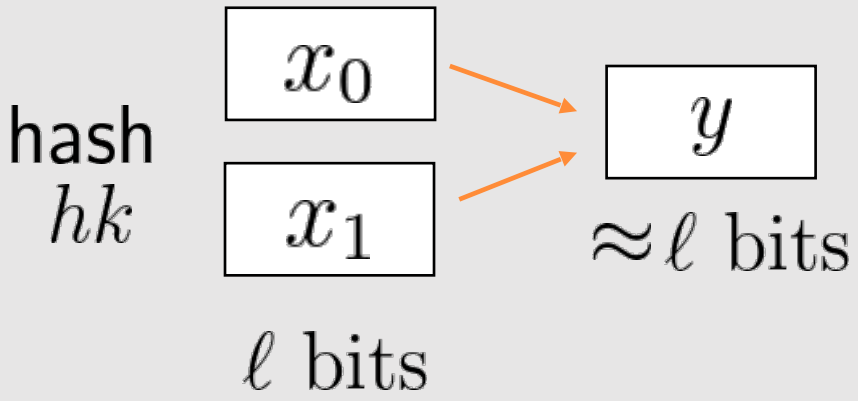
If $C(x, w) = 1$,
output MAC on (C, x)

Verify

If MAC verifies on (C, x)
Output 1

Ingredients

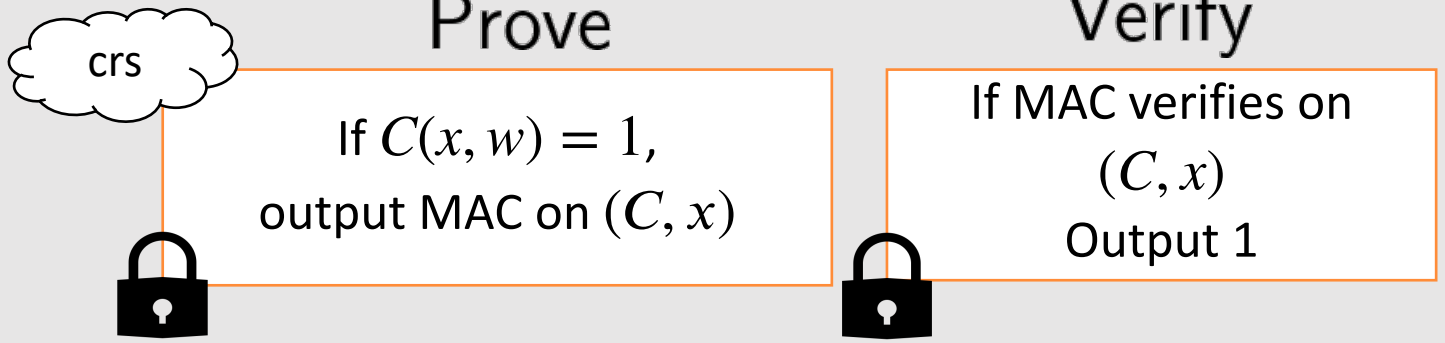
Two-to-one Hash function ★



★ **Statistical Binding Property (positional accumulator)**

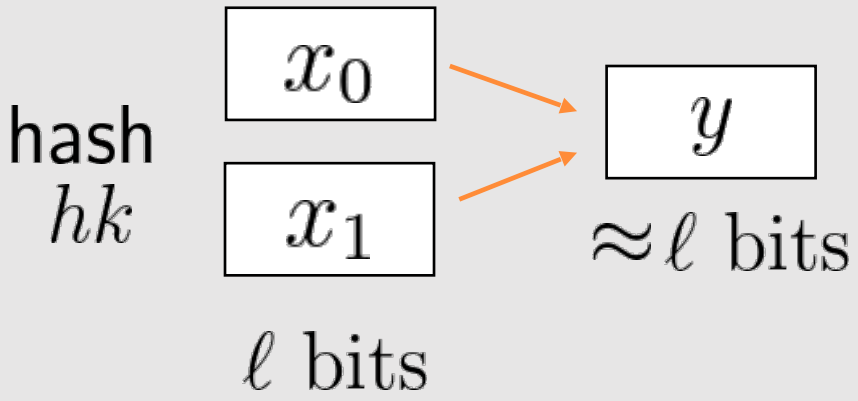
Indistinguishability Obfuscation + **One-Way functions**

SNARG for NP [SW14]



Ingredients

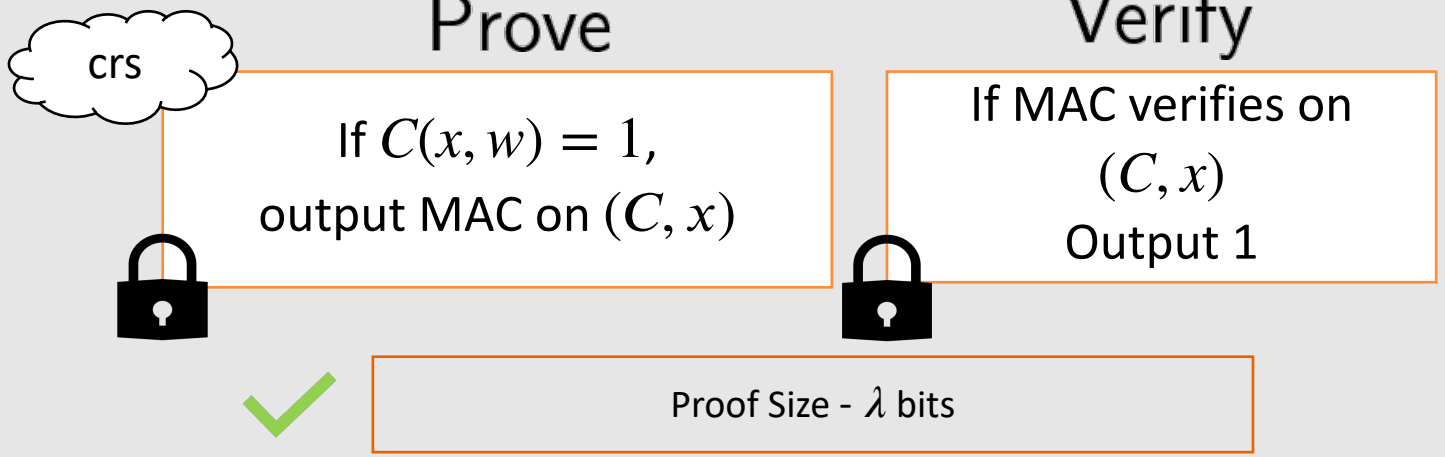
Two-to-one Hash function ★



★ **Statistical Binding Property (positional accumulator)**

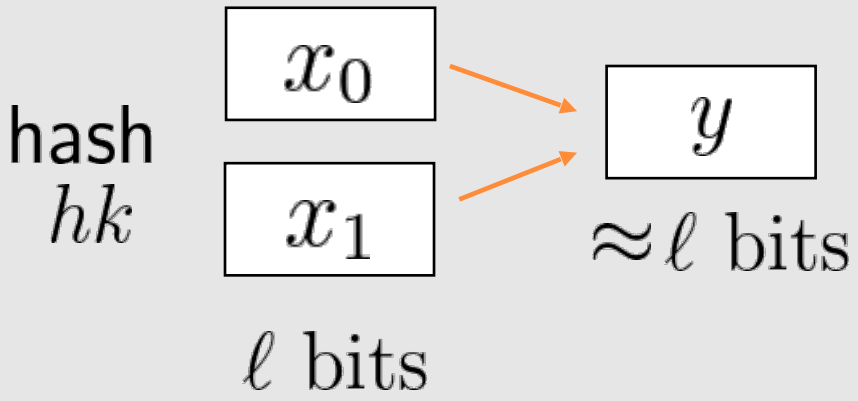
Indistinguishability Obfuscation + **One-Way functions**

SNARG for NP [SW14]



Ingredients

Two-to-one Hash function ★



★ **Statistical Binding Property (positional accumulator)**

Indistinguishability Obfuscation + **One-Way functions**

SNARG for NP [SW14]

crs

Prove

If $C(x, w) = 1$,
output MAC on (C, x)

Verify

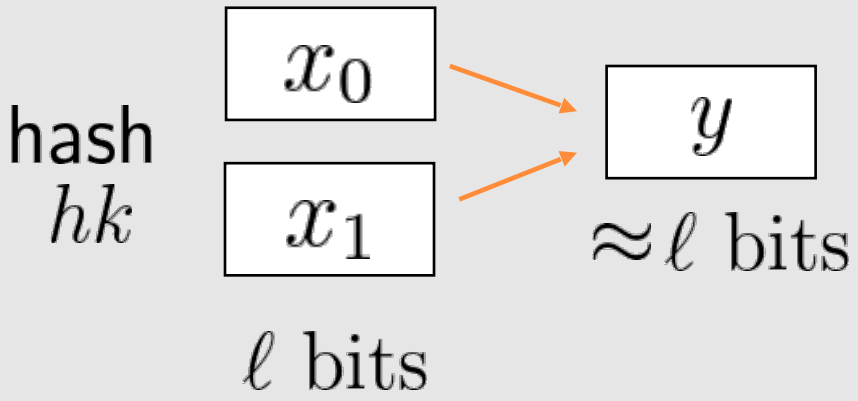
If MAC verifies on (C, x)
Output 1

✓ Proof Size - λ bits

✗ Support arbitrary polynomial instances

Ingredients

Two-to-one Hash function ★



★ **Statistical Binding Property (positional accumulator)**

Indistinguishability Obfuscation + **One-Way functions**

SNARG for NP [SW14]

crs

Prove

If $C(x, w) = 1$,
output MAC on (C, x)

Verify

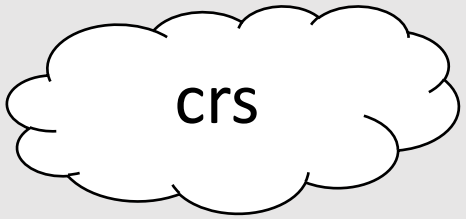
If MAC verifies on (C, x)
Output 1

✓ Proof Size - λ bits

✗ Support arbitrary polynomial instances

✗ Can update proofs!

Two-to-one proof merger



hk

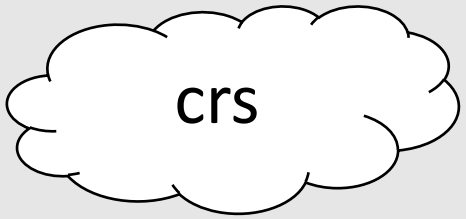
Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

If MAC verifies on $C(x_i, w_i)$,
Output 1

Two-to-one proof merger



hk

ProofMerge

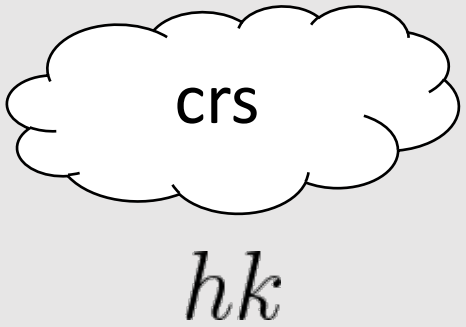
Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

If MAC verifies on $C(x_i, w_i)$,
Output 1

Two-to-one proof merger



ProofMerge

1. Verify MAC $-\pi_{left}$ on $(C, h_{left}, left)$

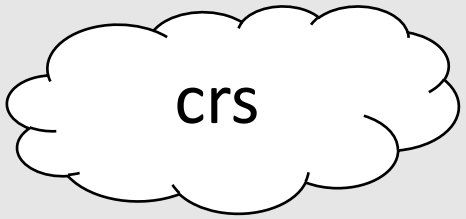
Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

If MAC verifies on $C(x_i, w_i)$,
Output 1

Two-to-one proof merger



hk

ProofMerge

1. Verify MAC $-\pi_{left}$ on $(C, h_{left}, left)$
2. Verify MAC $-\pi_{right}$ on $(C, h_{right}, right)$

$h_{left}\ \pi_{left}$ $h_{right}\ \pi_{right}$

Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

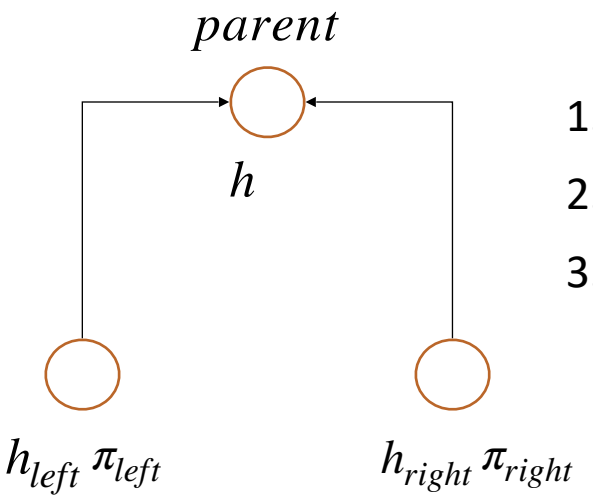
If MAC verifies on $C(x_i, w_i)$,
Output 1

Two-to-one proof merger

crs

hk

ProofMerge



1. Verify MAC $-\pi_{left}$ on $(C, h_{left}, left)$
2. Verify MAC $-\pi_{right}$ on $(C, h_{right}, right)$
3. Compute $h = hash(hk, h_{left}, h_{right})$

Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

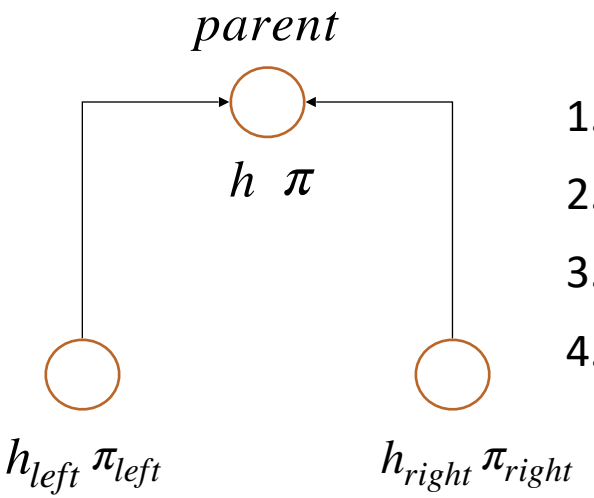
If MAC verifies on $C(x_i, w_i)$,
Output 1

Two-to-one proof merger

crs

hk

ProofMerge



1. Verify MAC $-\pi_{left}$ on $(C, h_{left}, left)$
2. Verify MAC $-\pi_{right}$ on $(C, h_{right}, right)$
3. Compute $h = hash(hk, h_{left}, h_{right})$
4. Output MAC π on $(C, h, parent)$

Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

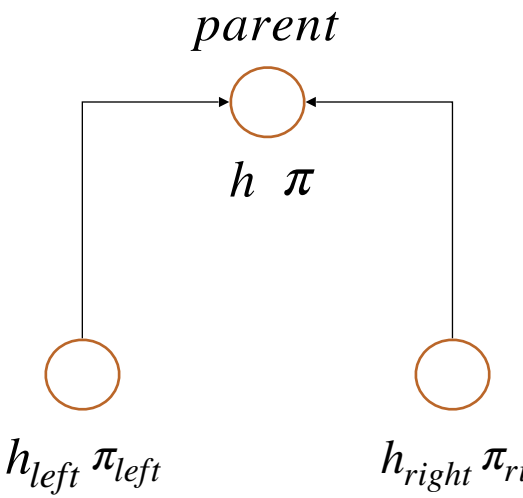
If MAC verifies on $C(x_i, w_i)$,
Output 1

Two-to-one proof merger

crs

hk

ProofMerge



1. Verify MAC $-\pi_{left}$ on $(C, h_{left}, left)$
2. Verify MAC $-\pi_{right}$ on $(C, h_{right}, right)$
3. Compute $h = hash(hk, h_{left}, h_{right})$
4. Output MAC π on $(C, h, parent)$

Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

If MAC verifies on $C(x_i, w_i)$,
Output 1

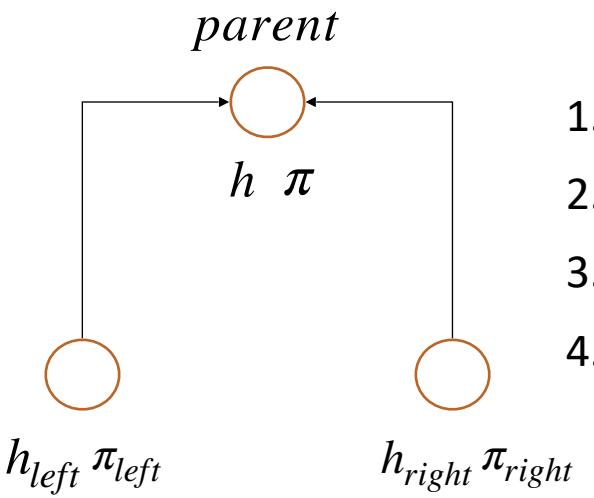
Two-to-one proof merger

crs

hk

Proof Size λ $bits$

ProofMerge



1. Verify MAC $-\pi_{left}$ on $(C, h_{left}, left)$
2. Verify MAC $-\pi_{right}$ on $(C, h_{right}, right)$
3. Compute $h = hash(hk, h_{left}, h_{right})$
4. Output MAC π on $(C, h, parent)$

Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

If MAC verifies on $C(x_i, w_i)$,
Output 1

Two-to-one proof merger

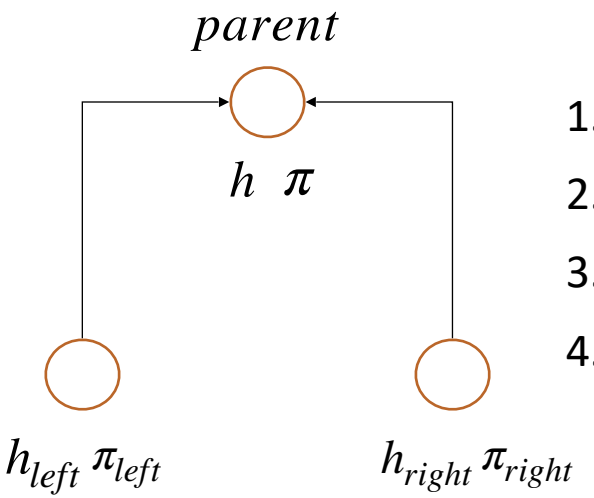
crs

hk

Proof Size λ $bits$

Support arbitrary polynomial instances

ProofMerge



1. Verify MAC $-\pi_{left}$ on $(C, h_{left}, left)$
2. Verify MAC $-\pi_{right}$ on $(C, h_{right}, right)$
3. Compute $h = hash(hk, h_{left}, h_{right})$
4. Output MAC π on $(C, h, parent)$

Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

If MAC verifies on $C(x_i, w_i)$,
Output 1

Two-to-one proof merger

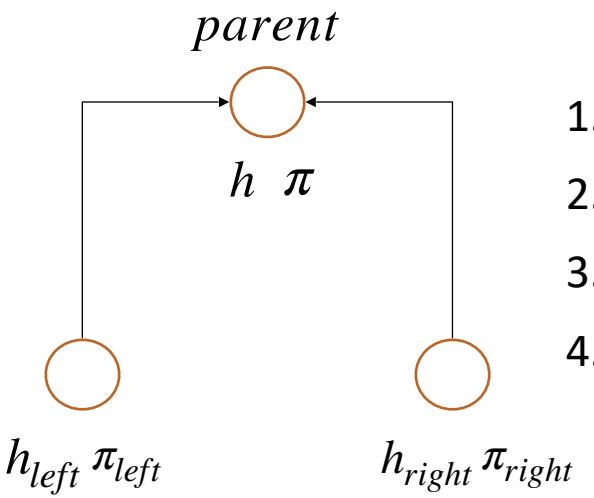
crs

hk

Proof Size λ $bits$

Support arbitrary polynomial instances

ProofMerge



1. Verify MAC $-\pi_{left}$ on $(C, h_{left}, left)$
2. Verify MAC $-\pi_{right}$ on $(C, h_{right}, right)$
3. Compute $h = hash(hk, h_{left}, h_{right})$
4. Output MAC π on $(C, h, parent)$

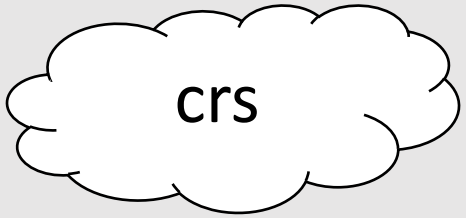
Prove

If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Verify

If MAC verifies on $C(x_i, w_i)$,
Output 1

Updatable BARG



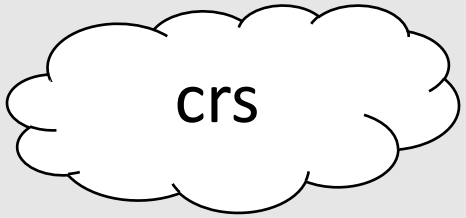
ProofMerge

Prove

Verify

hk

Updatable BARG

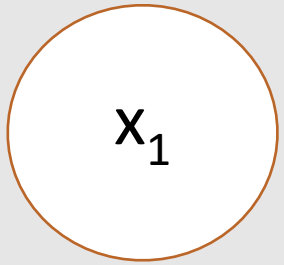


ProofMerge

Prove

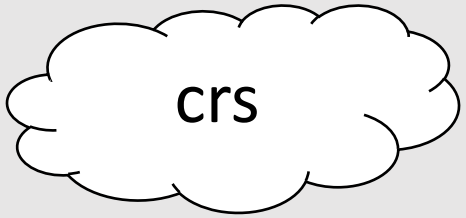
Verify

hk



w_1

Updatable BARG

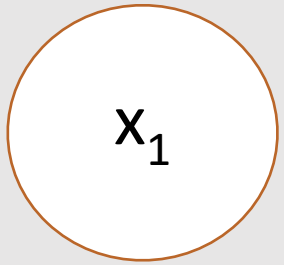


ProofMerge

Prove

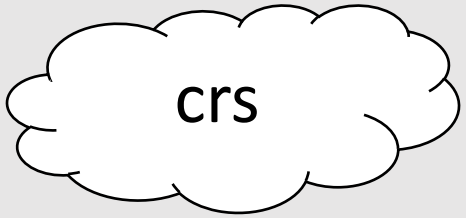
Verify

hk



$w_1\ \pi_1$

Updatable BARG

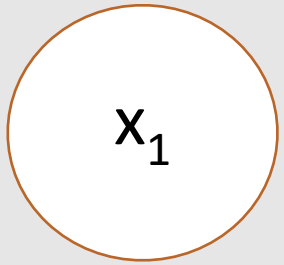


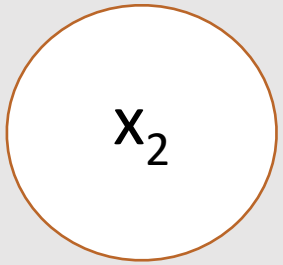
ProofMerge

Prove

Verify

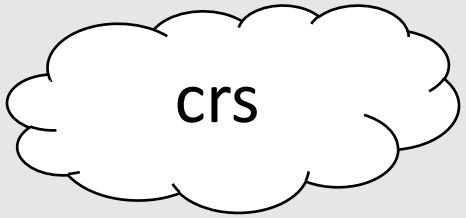
hk





$w_1\ \pi_1$ w_2

Updatable BARG

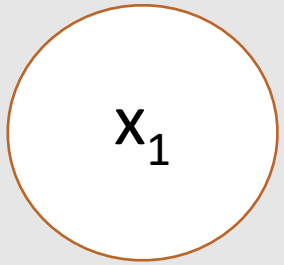


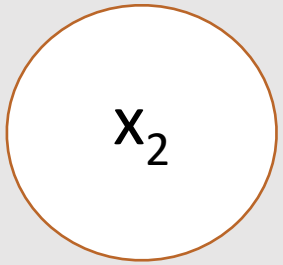
ProofMerge

Prove

Verify

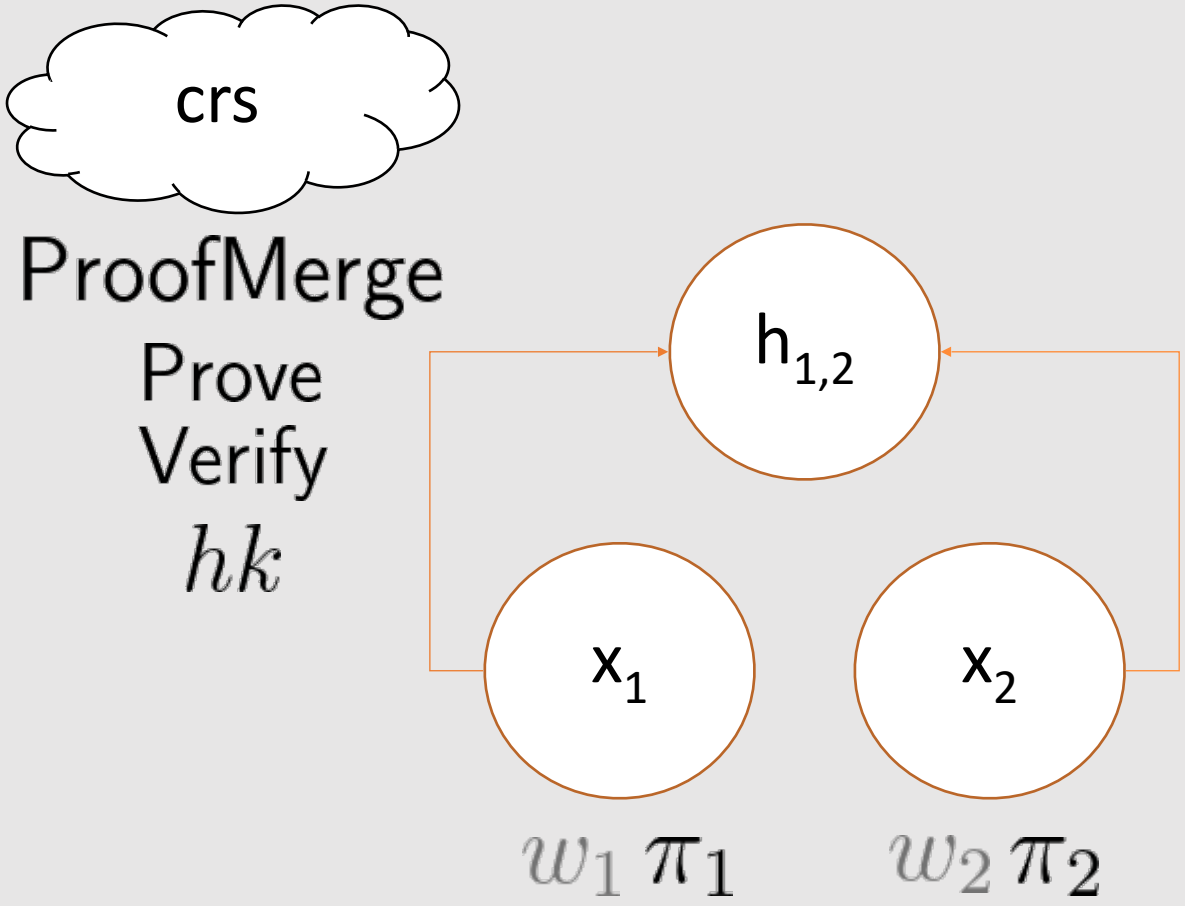
hk





$w_1\ \pi_1$ $w_2\ \pi_2$

Updatable BARG



Updatable BARG

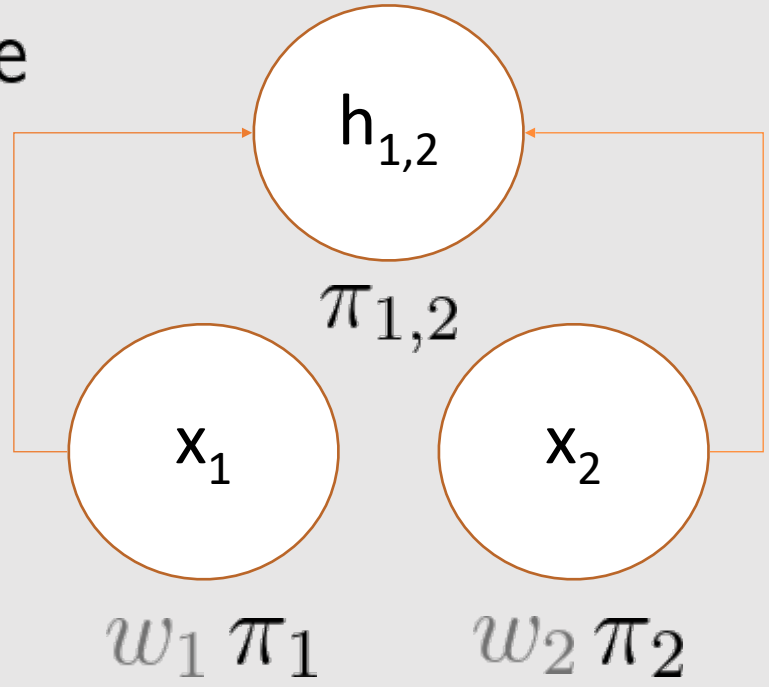
crs

ProofMerge

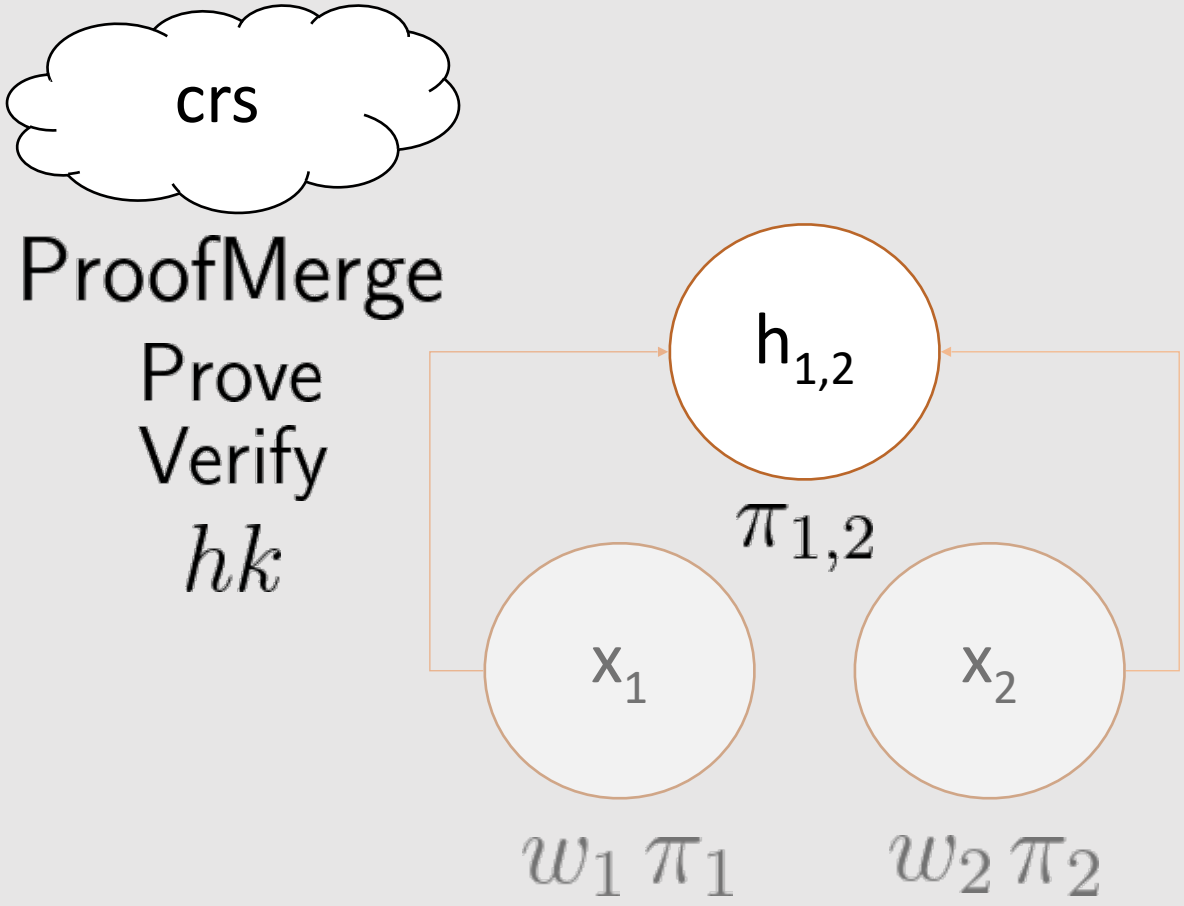
Prove

Verify

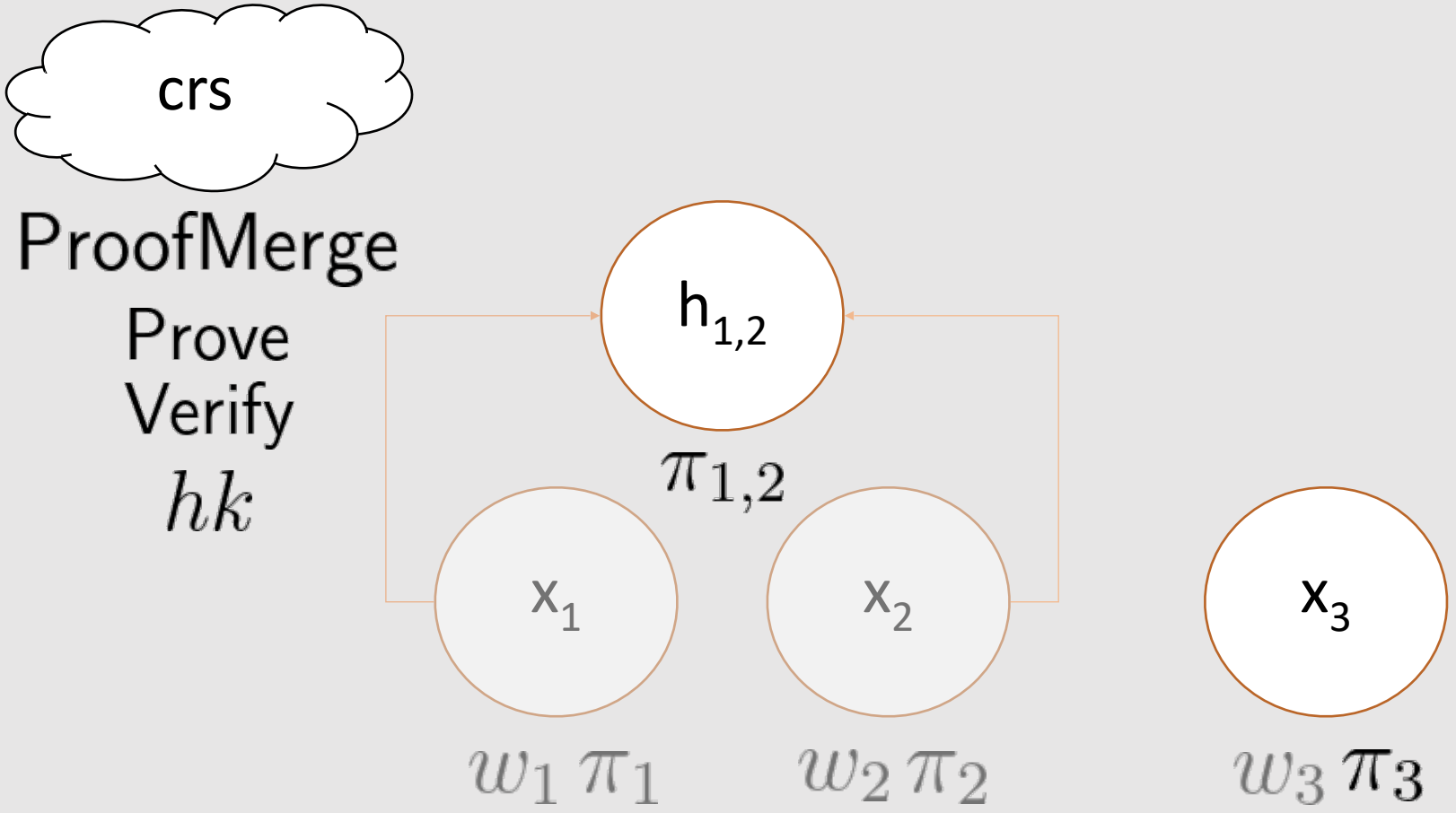
hk



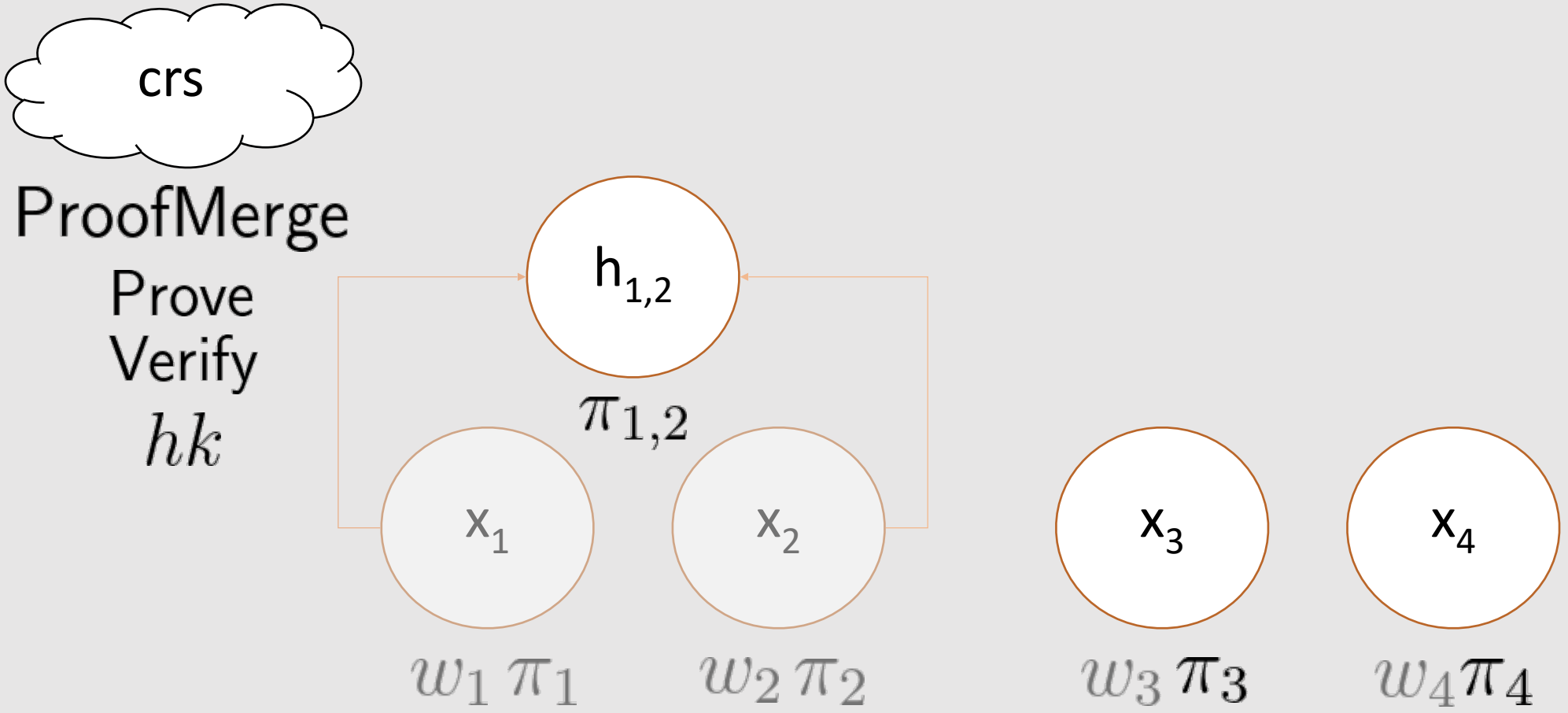
Updatable BARG



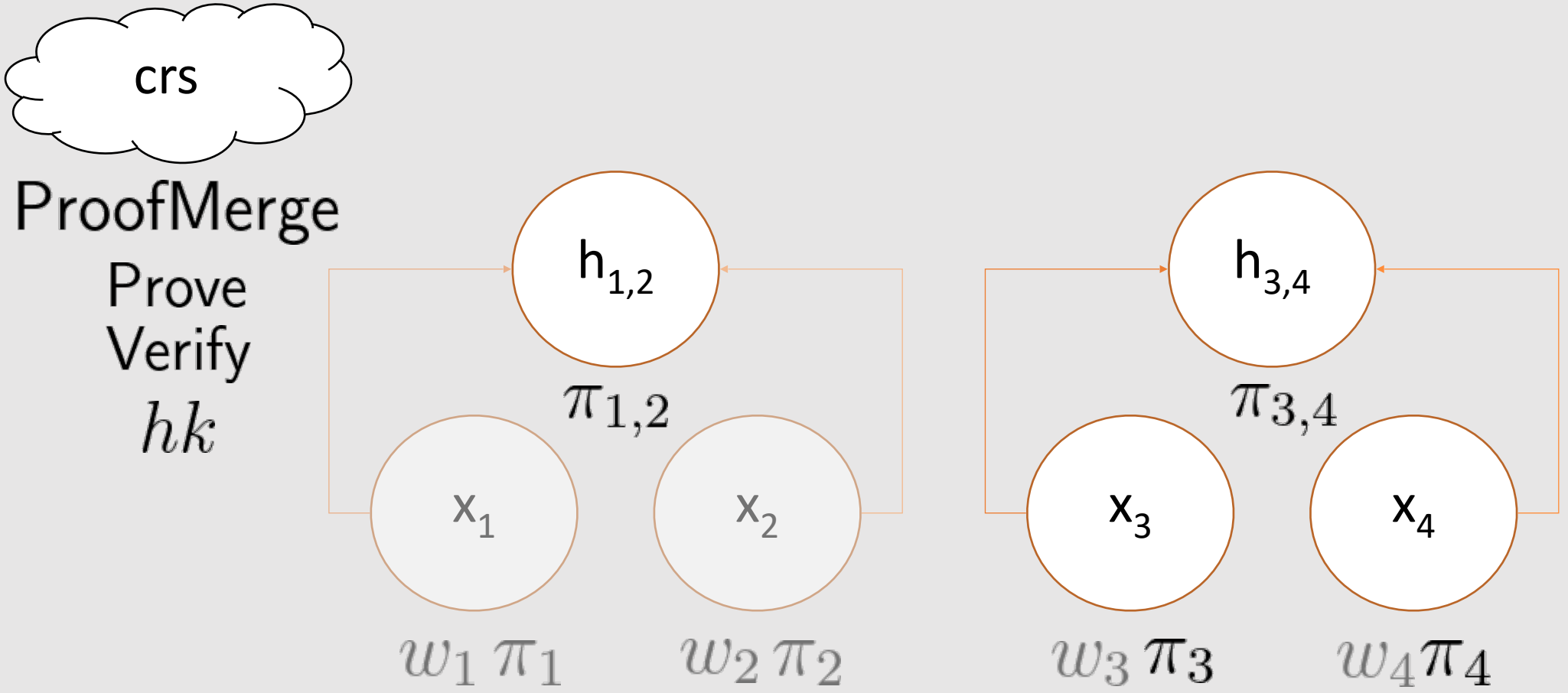
Updatable BARG



Updatable BARG



Updatable BARG



Updatable BARG

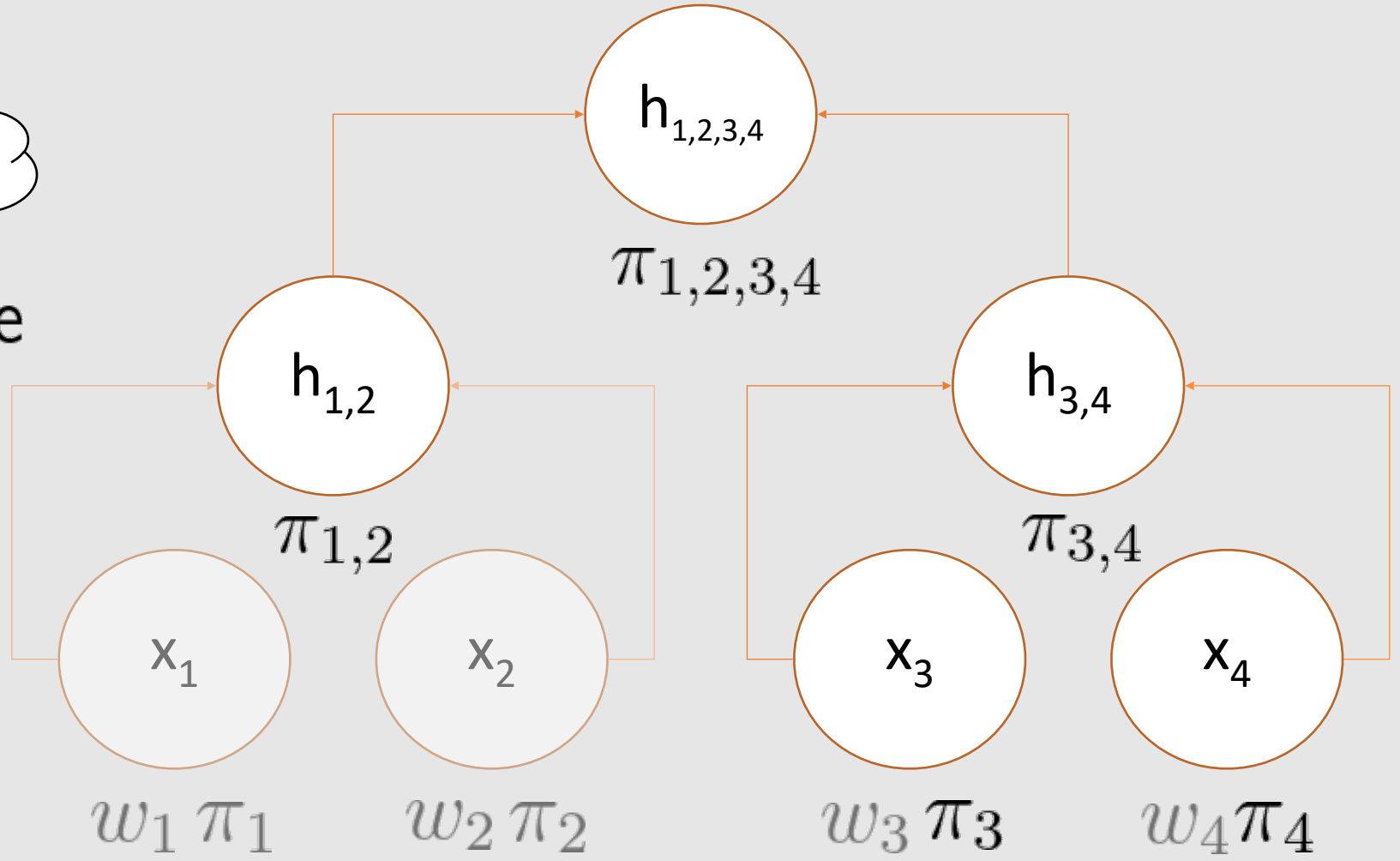
crs

ProofMerge

Prove

Verify

hk



Updatable BARG

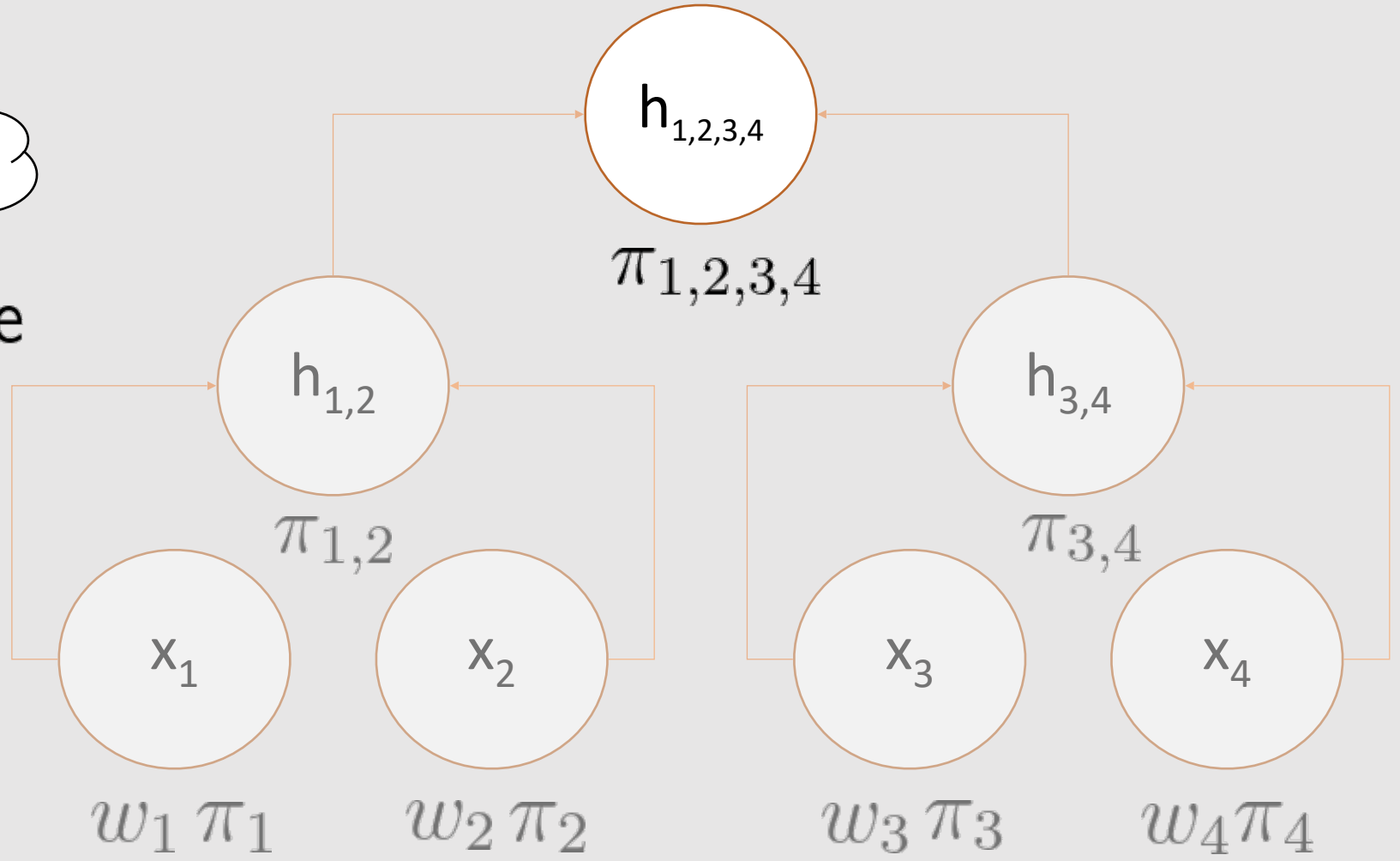
crs

ProofMerge

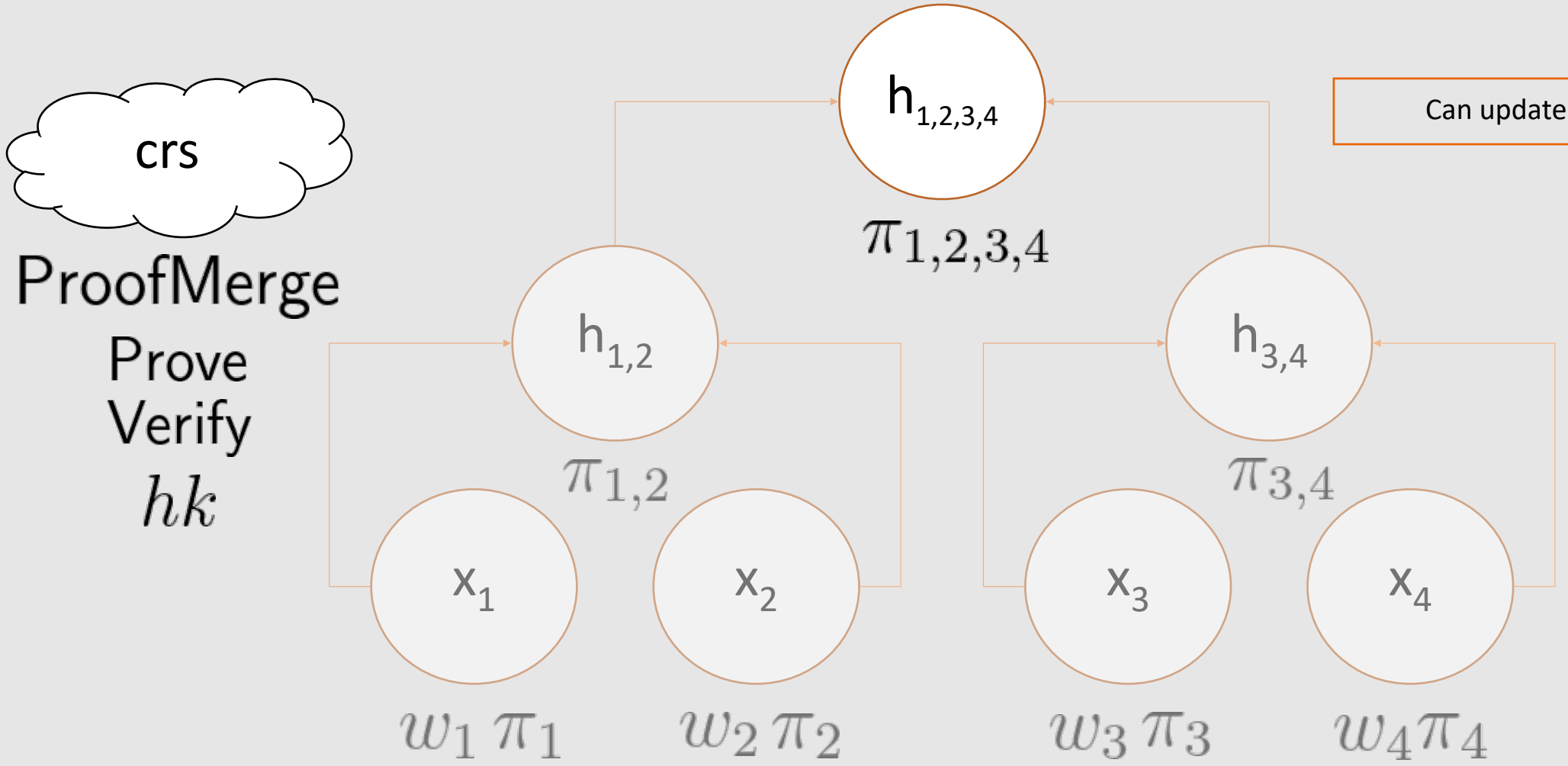
Prove

Verify

hk



Updatable BARG



Non-Adaptive Soundness

Non-Adaptive Soundness

$\exists i, x_i$ is not circuit satisfiable

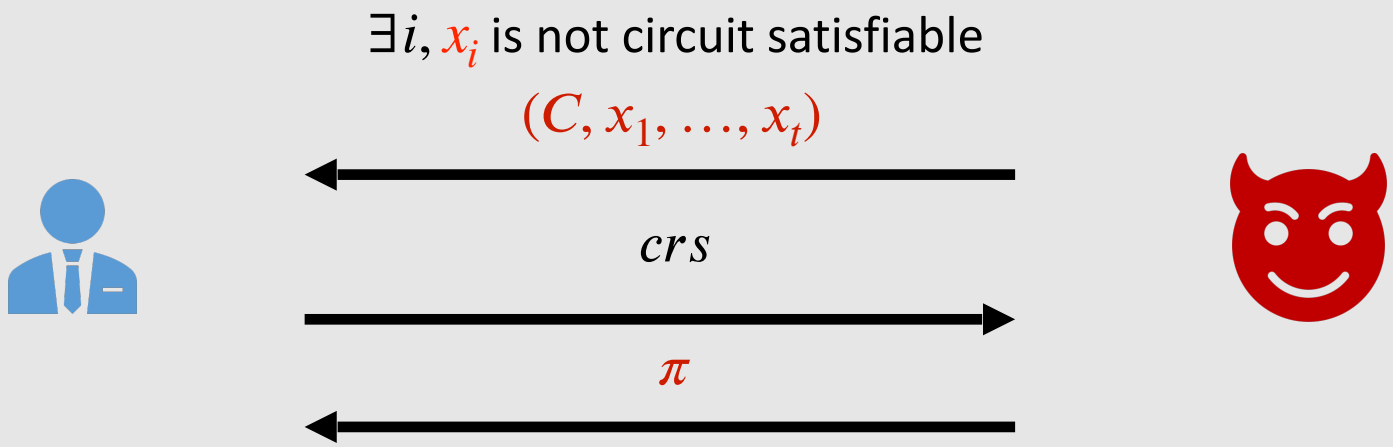
$(C, x_1, \ldots, x_t)$

crs

π

Outputs result of
Verify on proof π

Non-Adaptive Soundness



crs

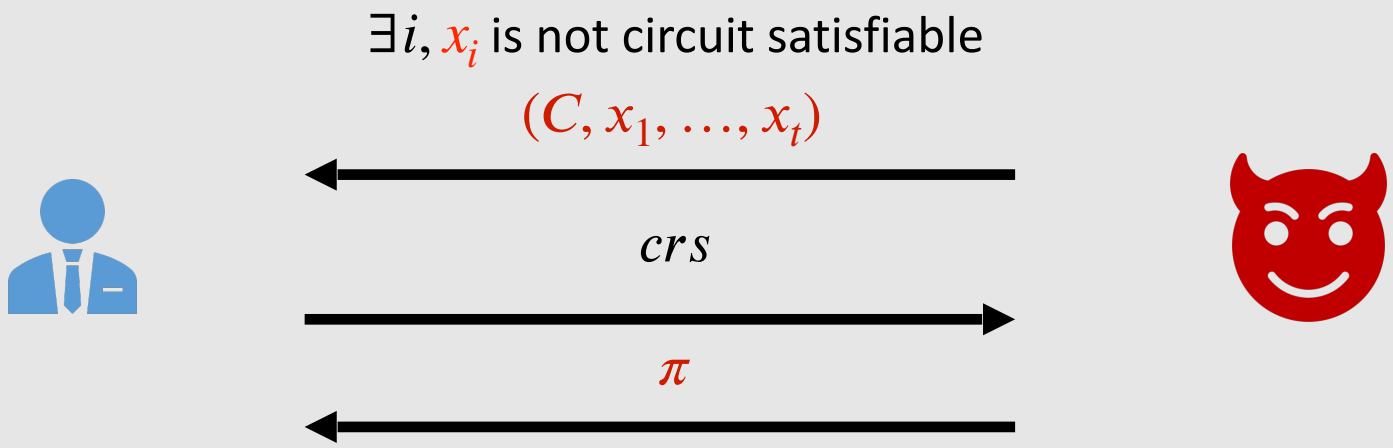
Prove

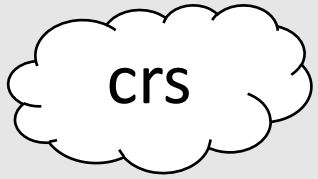
If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Prove

If $x = x_i$, then output $\perp$
If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

Non-Adaptive Soundness





Prove

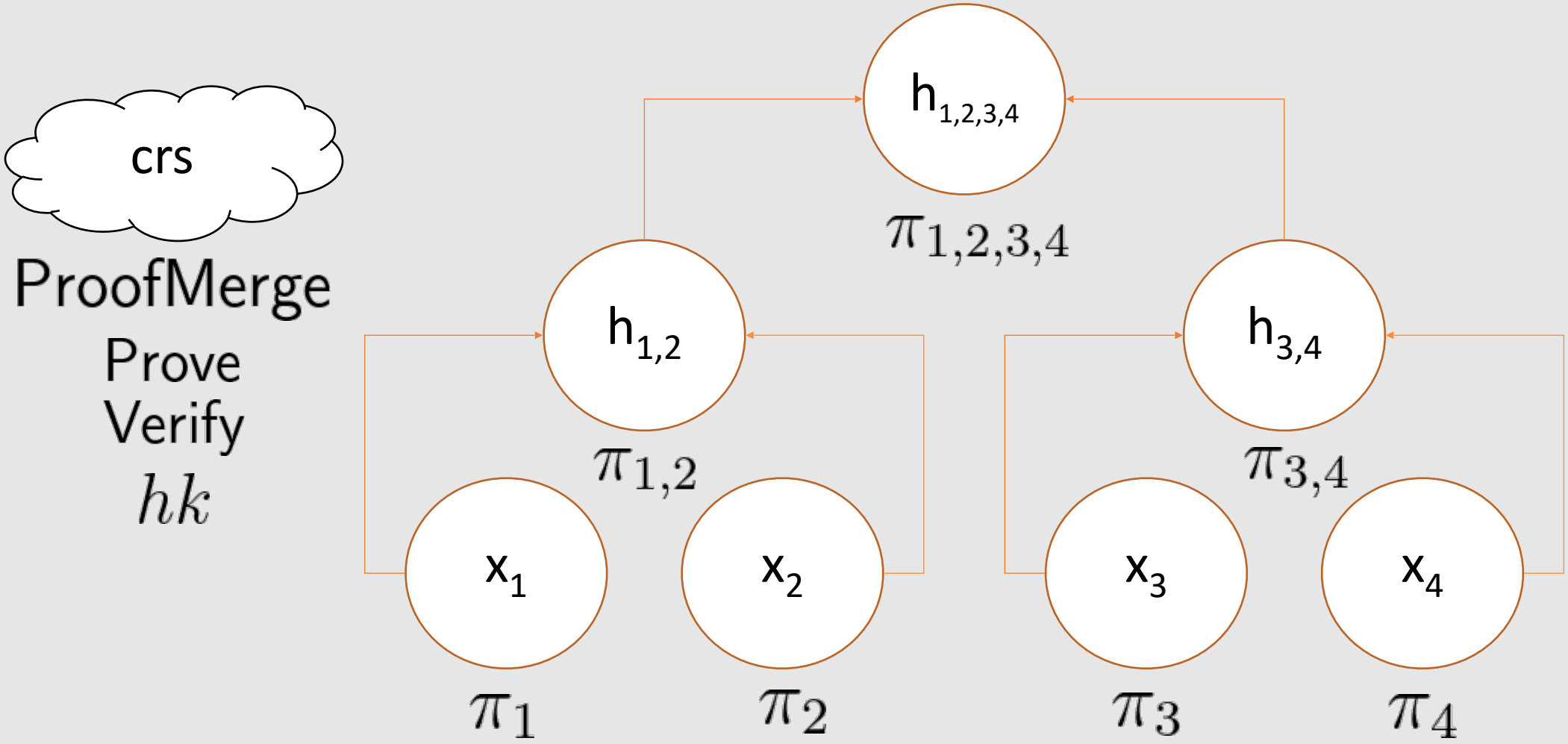
If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

$\approx_{iO}$

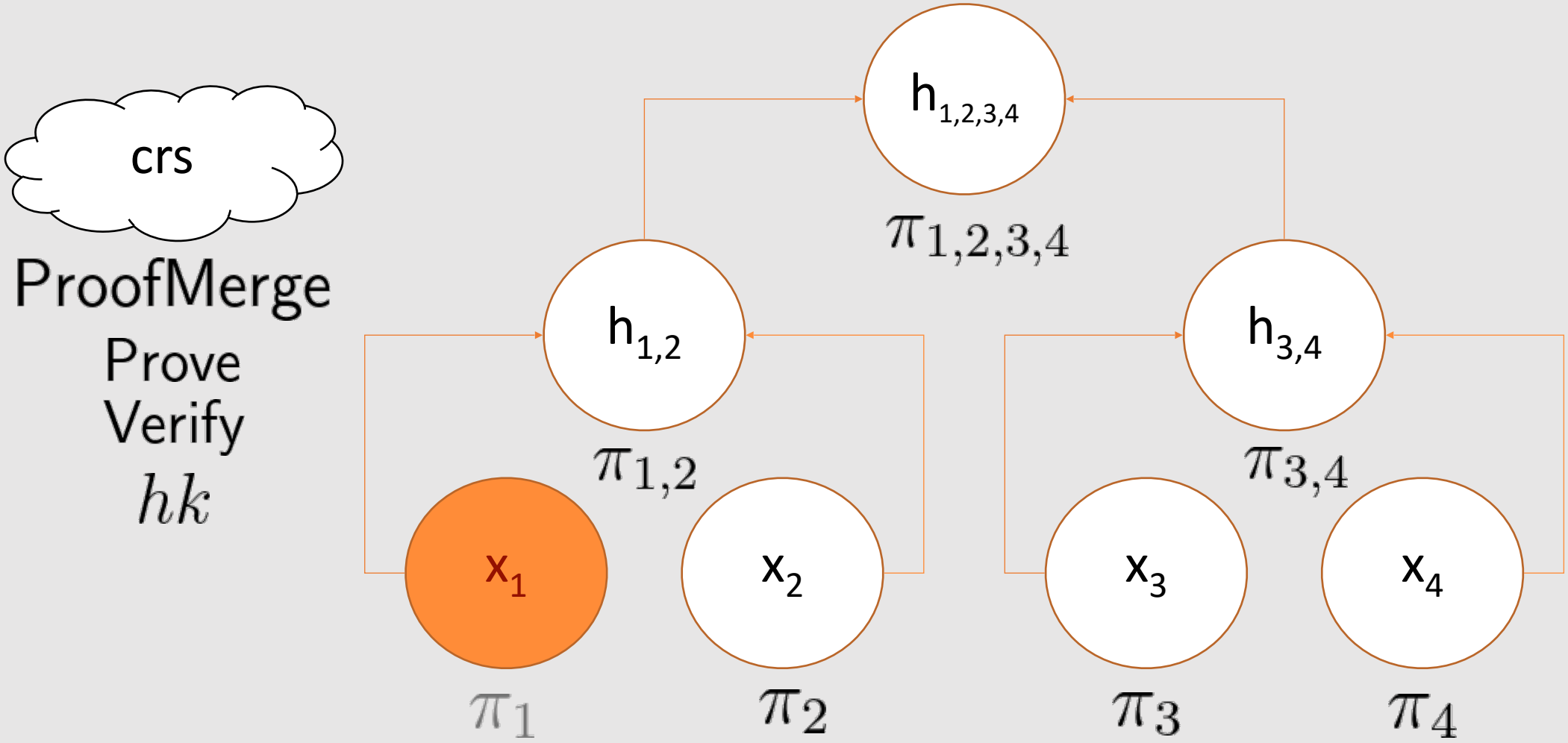
Prove

If $x = x_i$, then output $\perp$
If $C(x_i, w_i) = 1$,
output MAC on (C, x, i)

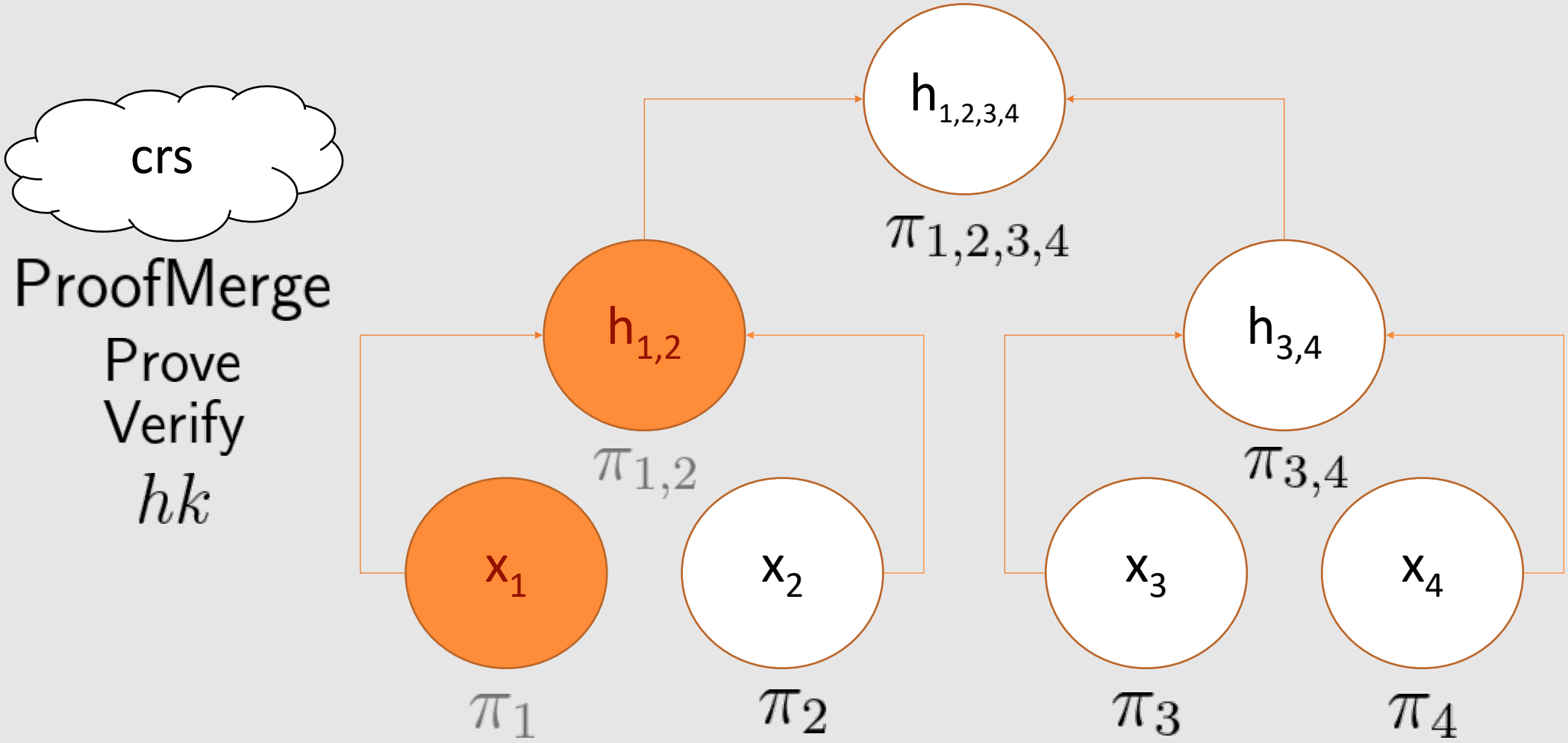
Non-Adaptive Soundness



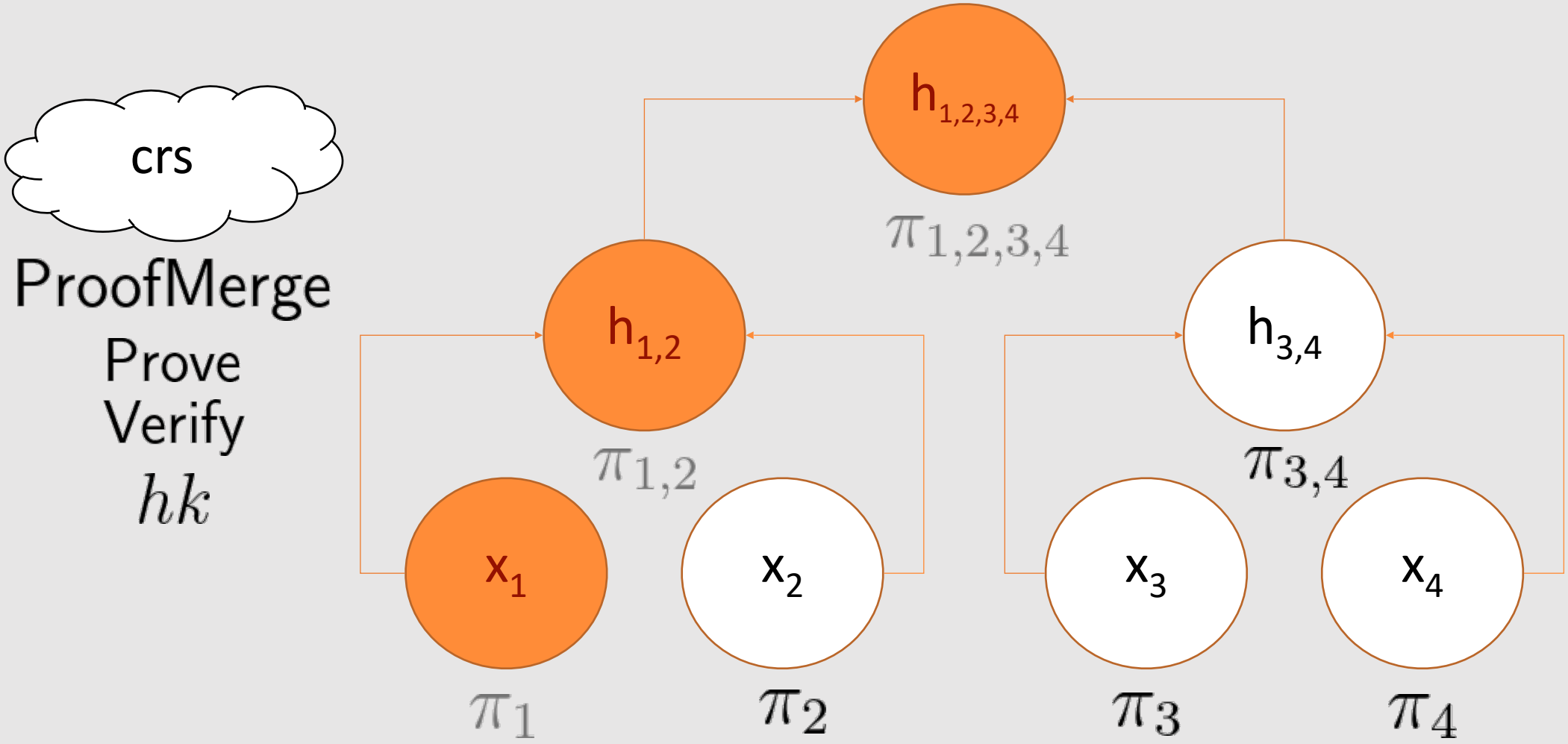
Non-Adaptive Soundness



Non-Adaptive Soundness



Non-Adaptive Soundness



BARGs for NP from iO

Support arbitrary poly instances

Proof size - $\lambda \cdot \log t$ bits

Can update proofs!

Non-adaptive soundness

A direct construction from iO+owf

BARGs for NP from iO

Support arbitrary poly instances

Proof size - $\lambda \cdot \log t$ bits

Can update proofs!

Non-adaptive soundness

Adaptive soundness

A direct construction from iO+owf

BARGs for NP from iO

Support arbitrary poly instances

Proof size - $\lambda \cdot \log t$ bits

Can update proofs!

Non-adaptive soundness

Adaptive soundness

A direct construction from iO+owf

Complexity leveraging
SSB hashing (LWE, DDH, DCR, φ -hiding)

BARGs for NP from iO

Support arbitrary poly instances

Proof size - $\lambda \cdot \log t$ bits

Can update proofs!

Non-adaptive soundness

A direct construction from iO+owf

Support arbitrary poly instances

Can update proofs!

Adaptive soundness

Complexity leveraging
SSB hashing (LWE, DDH, DCR, φ -hiding)

BARGs for NP from iO

Support arbitrary poly instances	Support arbitrary poly instances
Proof size - $\lambda \cdot \log t$ bits	Proof size – $\log t \cdot \text{poly}(\lambda, \lvert C \rvert)$ bits
Can update proofs!	Can update proofs!
Non-adaptive soundness	Adaptive soundness

A direct construction from iO+owf

Complexity leveraging
SSB hashing (LWE, DDH, DCR, φ -hiding)

BARGs for NP from iO

Support arbitrary poly instances	Support arbitrary poly instances
Proof size - $\lambda \cdot \log t$ bits	Proof size – $\log t \cdot \text{poly}(\lambda, \lvert C \rvert)$ bits
Can update proofs!	Can update proofs!
Non-adaptive soundness	Adaptive soundness
Zero Knowledge	Zero Knowledge
A direct construction from iO+owf	Complexity leveraging SSB hashing (LWE, DDH, DCR, φ -hiding)

Thanks!

Soon to be on eprint.iacr.org