batching adaptively-sound SNARGs for NP

Lali Devadas
MIT

joint work with Brent Waters (UT Austin and NTT Research) and David J. Wu (UT Austin)

Succinct Non-interactive ARGument (SNARG)

NP language L

common reference string

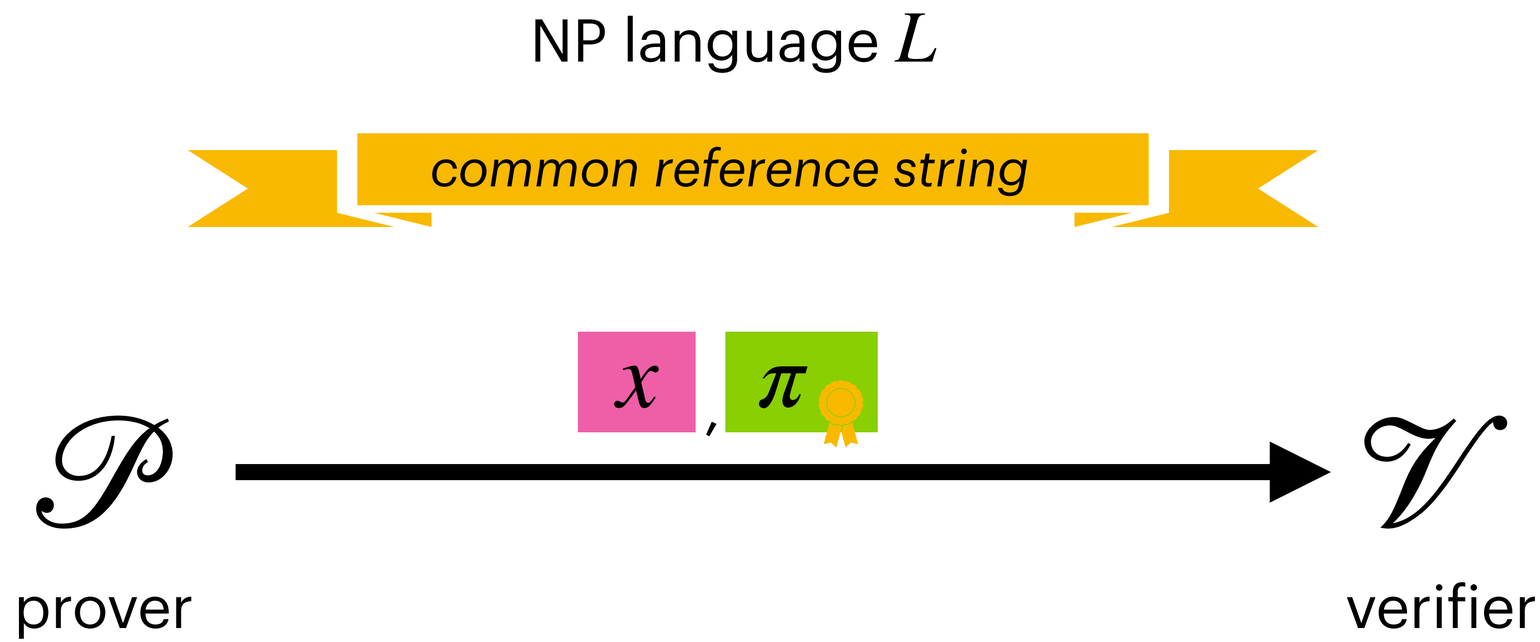
$\mathcal{P}$

prover

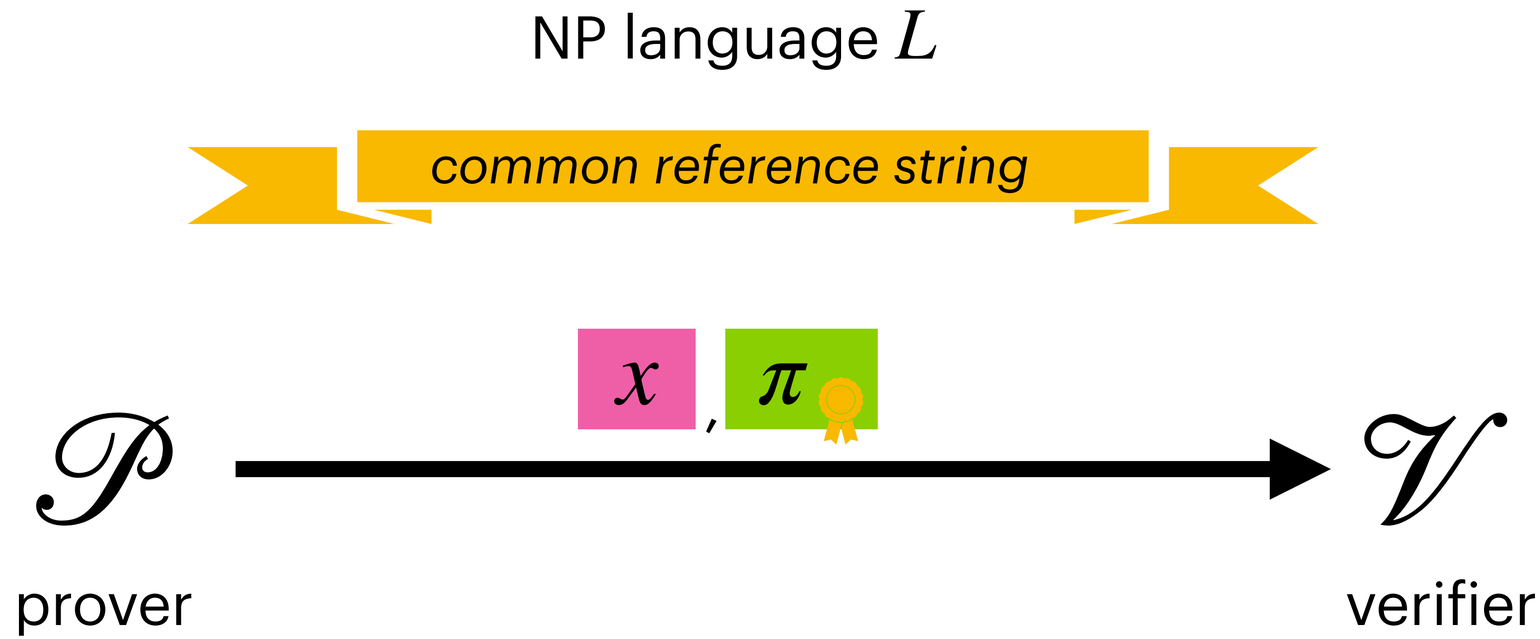
$\mathcal{V}$

verifier

Succinct Non-interactive ARGument (SNARG)

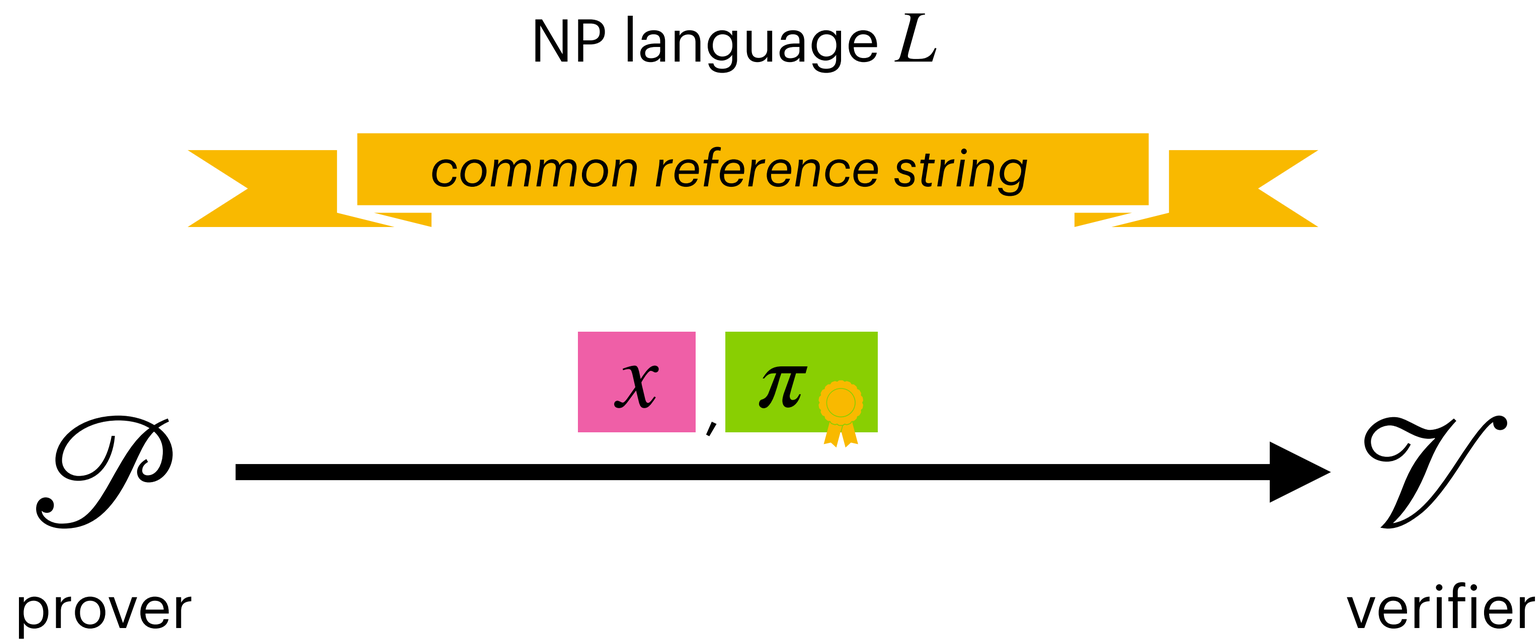


Succinct Non-interactive ARGument (SNARG)



completeness. honestly generated proofs are accepted.

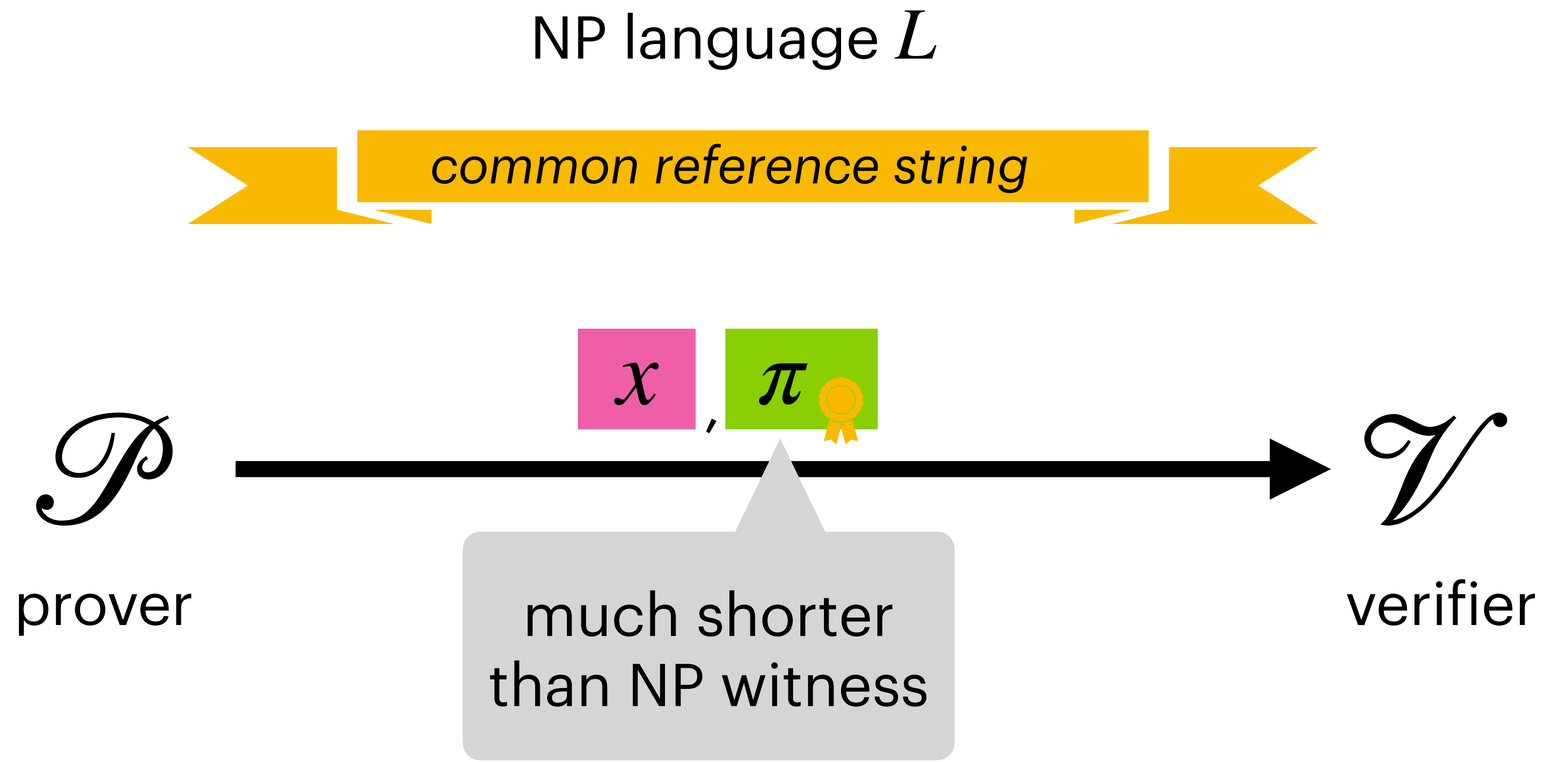
Succinct Non-interactive ARGument (SNARG)



completeness. honestly generated proofs are accepted.

(adaptive) soundness. no efficient adversary can fool the verifier.

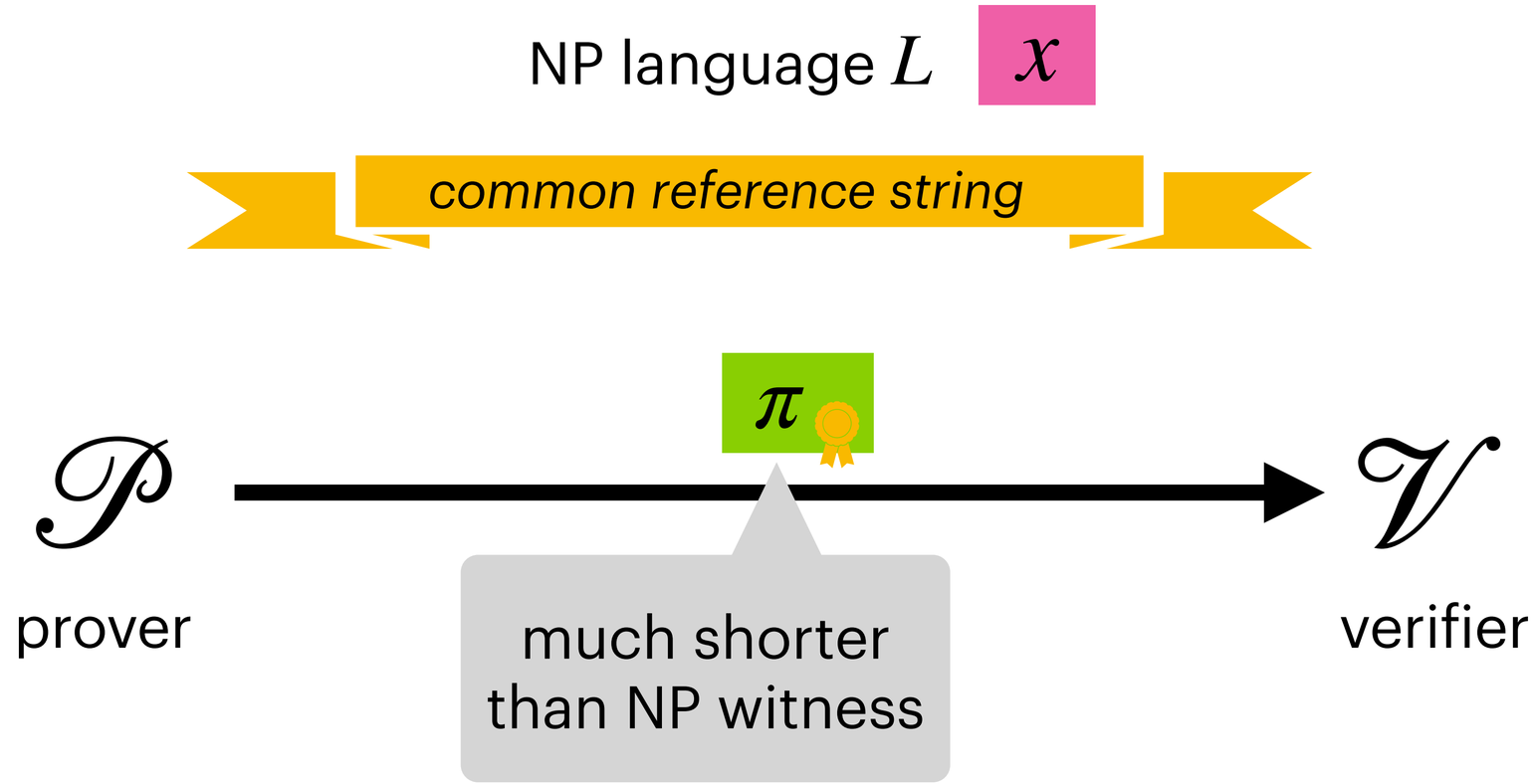
Succinct Non-interactive ARGument (SNARG)



completeness. honestly generated proofs are accepted.

(adaptive) soundness. no efficient adversary can fool the verifier.

Succinct Non-interactive ARGument (SNARG)



completeness. honestly generated proofs are accepted.

(non-adaptive) soundness. no efficient adversary can fool the verifier.

state of SNARGs

state of SNARGs

- constructions in idealized models
  - random oracle model [Mic94, Val08, BCS16, BBHR19, CMS19, COS20, CY21, ...]
  - generic/algebraic group model [Gro16, GWC19, MBKM19, CHMMVW20, Lip24, DMS24, ...]

state of SNARGs

- constructions in idealized models
  - random oracle model [Mic94, Val08, BCS16, BBHR19, CMS19, COS20, CY21, ...]
  - generic/algebraic group model [Gro16, GWC19, MBKM19, CHMMVW20, Lip24, DMS24, ...]
- constructions from knowledge assumptions: [Gro10, BCCT12, GGPR13, BCIOP13, BCPR14, BISW17, ACLMT22, CLM23, ...]

state of SNARGs

- constructions in idealized models
  - random oracle model [Mic94, Val08, BCS16, BBHR19, CMS19, COS20, CY21, ...]
  - generic/algebraic group model [Gro16, GWC19, MBKM19, CHMMVW20, Lip24, DMS24, ...]
- constructions from knowledge assumptions: [Gro10, BCCT12, GGPR13, BCIOP13, BCPR14, BISW17, ACLMT22, CLM23, ...]

what about from falsifiable assumptions?

state of SNARGs

- constructions in idealized models
  - random oracle model [Mic94, Val08, BCS16, BBHR19, CMS19, COS20, CY21, ...]
  - generic/algebraic group model [Gro16, GWC19, MBKM19, CHMMVW20, Lip24, DMS24, ...]
- constructions from knowledge assumptions: [Gro10, BCCT12, GGPR13, BCIOP13, BCPR14, BISW17, ACLMT22, CLM23, ...]

what about from falsifiable assumptions?

- Sahai-Waters14: non-adaptively-sound SNARG for NP from indistinguishability obfuscation and one-way functions

state of SNARGs

- constructions in idealized models
  - random oracle model [Mic94, Val08, BCS16, BBHR19, CMS19, COS20, CY21, ...]
  - generic/algebraic group model [Gro16, GWC19, MBKM19, CHMMVW20, Lip24, DMS24, ...]
- constructions from knowledge assumptions: [Gro10, BCCT12, GGPR13, BCIOP13, BCPR14, BISW17, ACLMT22, CLM23, ...]

what about from falsifiable assumptions?

- Sahai-Waters14: non-adaptively-sound SNARG for NP from indistinguishability obfuscation and one-way functions
- Jain-Lin-Sahai21-22: indistinguishability obfuscation from falsifiable assumptions

state of SNARGs

- constructions in idealized models
  - random oracle model [Mic94, Val08, BCS16, BBHR19, CMS19, COS20, CY21, ...]
  - generic/algebraic group model [Gro16, GWC19, MBKM19, CHMMVW20, Lip24, DMS24, ...]
- constructions from knowledge assumptions: [Gro10, BCCT12, GGPR13, BCIOP13, BCPR14, BISW17, ACLMT22, CLM23, ...]

what about from falsifiable assumptions?

- Sahai-Waters14: non-adaptively-sound SNARG for NP from indistinguishability obfuscation and one-way functions
- Jain-Lin-Sahai21-22: indistinguishability obfuscation from falsifiable assumptions

what about adaptive soundness?

Gentry-Wichs separation

(informal) theorem. there does not exist an adaptively-sound SNARG for NP with a black-box security reduction to falsifiable assumptions.

Gentry-Wichs separation

(informal) theorem. there does not exist an adaptively-sound SNARG for NP with a black-box security reduction to falsifiable assumptions.

- does **not** rule out reductions which run in enough time to decide the NP language!

Gentry-Wichs separation

(informal) theorem. there does not exist an adaptively-sound SNARG for NP with a black-box security reduction to falsifiable assumptions.

- does **not** rule out reductions which run in enough time to decide the NP language!
- **one interpretation**: the CRS for a language decidable in time T must at least grow with $\mathsf{polylog}(T)$ to keep the proof succinct

adaptively-sound SNARGs for NP

from falsifiable assumptions

iO = indistinguishability obfuscation
OWF = one-way functions

adaptively-sound SNARGs for NP

from falsifiable assumptions

- [WW24a]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF and rerandomizable OWF (e.g., from discrete log / factoring)

iO = indistinguishability obfuscation
OWF = one-way functions

adaptively-sound SNARGs for NP

from falsifiable assumptions

- [WW24a]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF and rerandomizable OWF (e.g., from discrete log / factoring)

- [MPV24]: Sahai-Waters SNARG for NP from subexponentially-secure iO+OWF is adaptively sound in the designated-verifier model

iO = indistinguishability obfuscation
OWF = one-way functions

adaptively-sound SNARGs for NP

from falsifiable assumptions

- [WW24a]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF and rerandomizable OWF (e.g., from discrete log / factoring)

- [MPV24]: Sahai-Waters SNARG for NP from subexponentially-secure iO+OWF is adaptively sound in the designated-verifier model

- [WZ24]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF and lossy functions (e.g., from learning with errors)

iO = indistinguishability obfuscation
OWF = one-way functions

adaptively-sound SNARGs for NP

from falsifiable assumptions

- [WW24a]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF and rerandomizable OWF (e.g., from discrete log / factoring)
- [MPV24]: Sahai-Waters SNARG for NP from subexponentially-secure iO+OWF is adaptively sound in the designated-verifier model
- [WZ24]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF and lossy functions (e.g., from learning with errors)
- [WW24b]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF

iO = indistinguishability obfuscation
OWF = one-way functions

adaptively-sound SNARGs for NP

from falsifiable assumptions

- [WW24a]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF and rerandomizable OWF (e.g., from discrete log / factoring)
- [MPV24]: Sahai-Waters SNARG for NP from subexponentially-secure iO+OWF is adaptively sound in the designated-verifier model
- [WZ24]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF and lossy functions (e.g., from learning with errors)
- [WW24b]: adaptively-sound SNARG for NP from subexponentially-secure iO+OWF

in all the above, the CRS grows with the size of the NP verification circuit

iO = indistinguishability obfuscation
OWF = one-way functions

can we get a short(ish) CRS?

can we get a short(ish) CRS?

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \; \forall i \in [m]$$

- $C =$ NP verifier for L

can we get a short(ish) CRS?

consider batch-NP, i.e., deciding whether

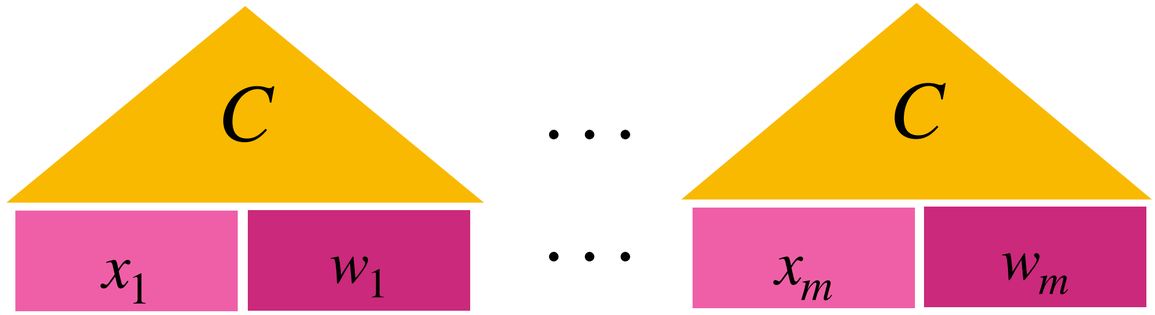
$$x_1, \ldots, x_m \in L \; \forall i \in [m]$$

- C = NP verifier for L

can we get a short(ish) CRS?

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \; \forall i \in [m]$$

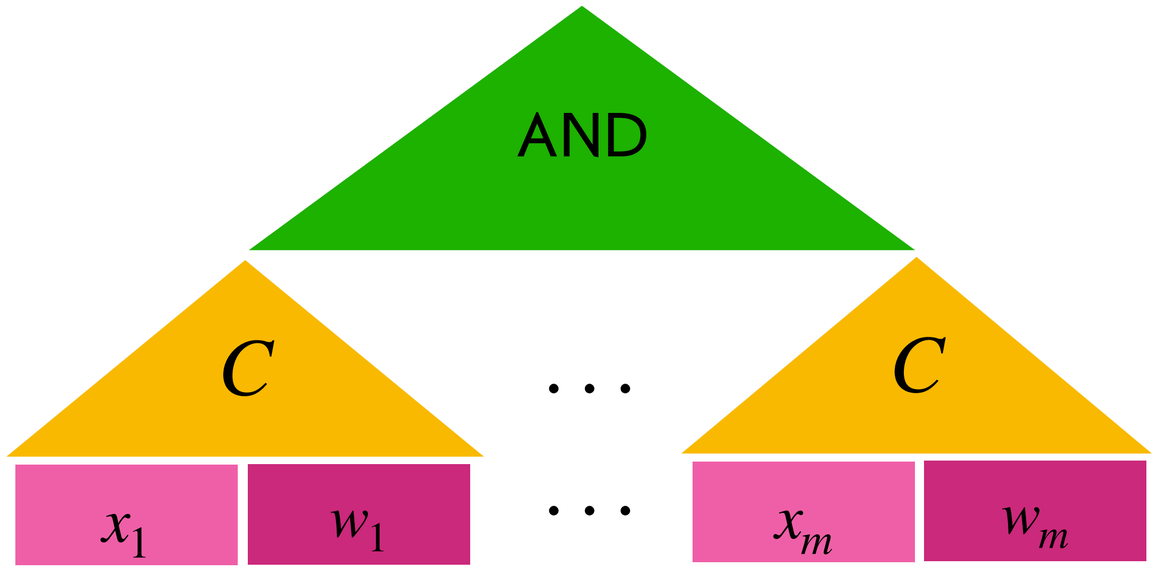


- C = NP verifier for L

can we get a short(ish) CRS?

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \; \forall i \in [m]$$



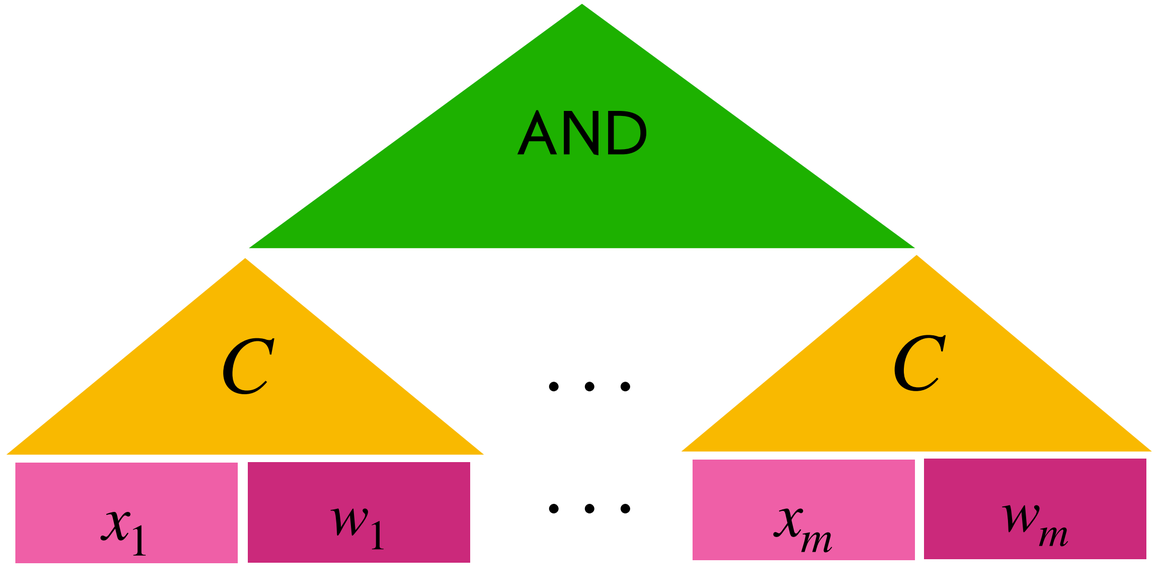
- C = NP verifier for L

can we get a short(ish) CRS?

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \; \forall i \in [m]$$

the NP verification circuit for batch-L has size $\approx |C| \cdot m$



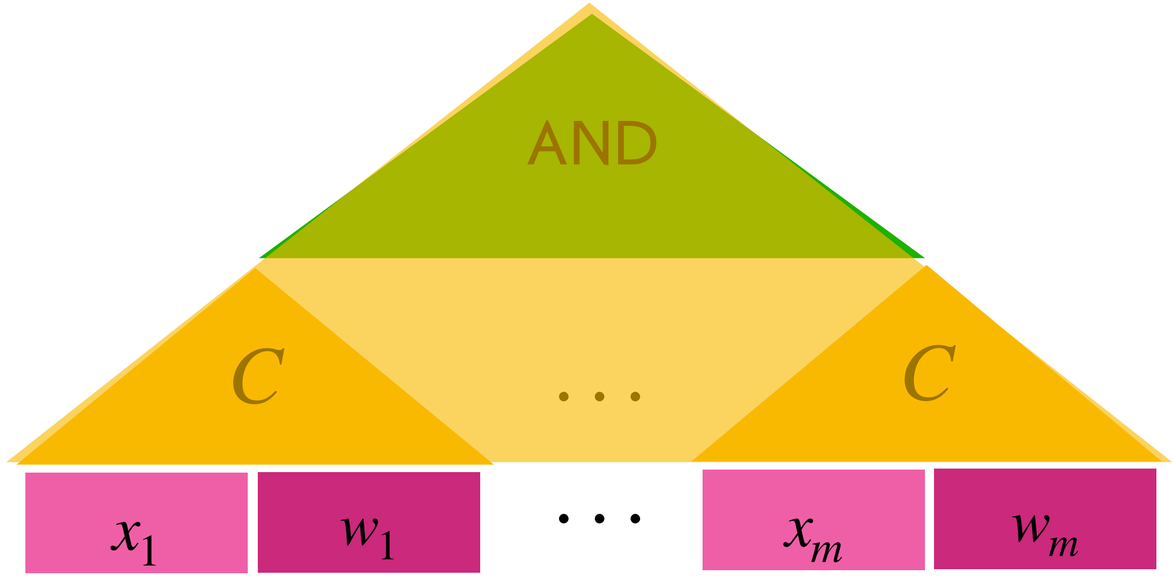
- C = NP verifier for L

can we get a short(ish) CRS?

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \; \forall i \in [m]$$

the NP verification circuit for batch-L has size $\approx |C| \cdot m$



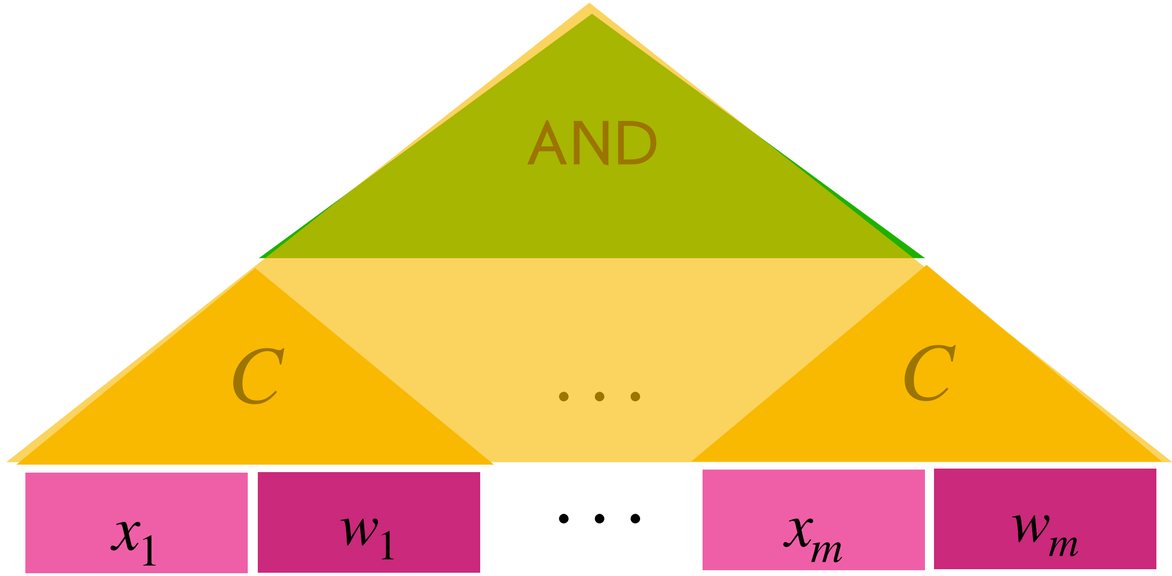
- C = NP verifier for L

can we get a short(ish) CRS?

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \; \forall i \in [m]$$

the NP verification circuit for batch-L has size $\approx |C| \cdot m$



so if we use *any* existing adaptively-sound SNARG for batch-L, the CRS size will grow with $|C| \cdot m$...

can we get a short(ish) CRS?

- C = NP verifier for L
- T = time to decide L

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \ \forall i \in [m]$$

existing adaptively-sound SNARGs applied to L have a CRS that grows with $|C| \cdot m$...

can we get a short(ish) CRS?

- C = NP verifier for L
- T = time to decide L

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \; \forall i \in [m]$$

existing adaptively-sound SNARGs applied to L have a CRS that grows with $|C| \cdot m$...

but Gentry-Wichs only tells us that the CRS must grow with polylog$(m \cdot T)$ to keep the proof succinct...

time to decide L

- C = NP verifier for L
- T = time to decide L

can we get a short(ish) CRS?

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \ \forall i \in [m]$$

existing adaptively-sound SNARGs applied to L have a CRS that grows with $|C| \cdot m$...

but Gentry-Wichs only tells us that the CRS must grow with $\text{polylog}(m \cdot T)$ to keep the proof succinct...

time to decide L

can we construct an adaptively-sound SNARG for batch-NP where the CRS grows with $|C| \cdot \log m$ instead of $|C| \cdot m$?

from falsifiable assumptions

- C = NP verifier for L
- T = time to decide L

can we get a short(ish) CRS?

consider batch-NP, i.e., deciding whether

$$x_1, \ldots, x_m \in L \; \forall i \in [m]$$

existing adaptively-sound SNARGs applied to L have a CRS that grows with $|C| \cdot m$...

but Gentry-Wichs only tells us that the CRS must grow with $\text{polylog}(m \cdot T)$ to keep the proof succinct...

time to decide L

can we construct an adaptively-sound SNARG for batch-NP where the CRS grows with $|C| \cdot \log m$ instead of $|C| \cdot m$?

from falsifiable assumptions

this work: **yes!**

this work

theorem. there exists an adaptively-sound SNARG for batch-NP with CRS size $\mathrm{poly}(\lambda, |C|, \log m)$ and proof size $\mathrm{poly}(\lambda)$.

this work

theorem. there exists an adaptively-sound SNARG for batch-NP with CRS size poly$(\lambda, |C|, \log m)$ **and proof size** poly(λ).

- our construction uses (subexponentially hard) iO+OWF and a (statistically) rerandomizable PRG

this work

theorem. there exists an adaptively-sound SNARG for batch-NP with CRS size $\mathrm{poly}(\lambda, |C|, \log m)$ **and proof size** $\mathrm{poly}(\lambda)$**.**

- our construction uses (subexponentially hard) iO+OWF and a (statistically) rerandomizable PRG
- we instantiate the rerandomizable PRG from DDH

this work

theorem. there exists an adaptively-sound SNARG for batch-NP with CRS size $\mathrm{poly}(\lambda, |C|, \log m)$ **and proof size** $\mathrm{poly}(\lambda)$**.**

- our construction uses (subexponentially hard) iO+OWF and a (statistically) rerandomizable PRG
- we instantiate the rerandomizable PRG from DDH
- another proof of theorem assuming iO+OWF and a homomorphic rerandomizable OWF (instantiated from discrete log) in the paper

Sahai-Waters14:
non-adaptively-sound SNARG for NP

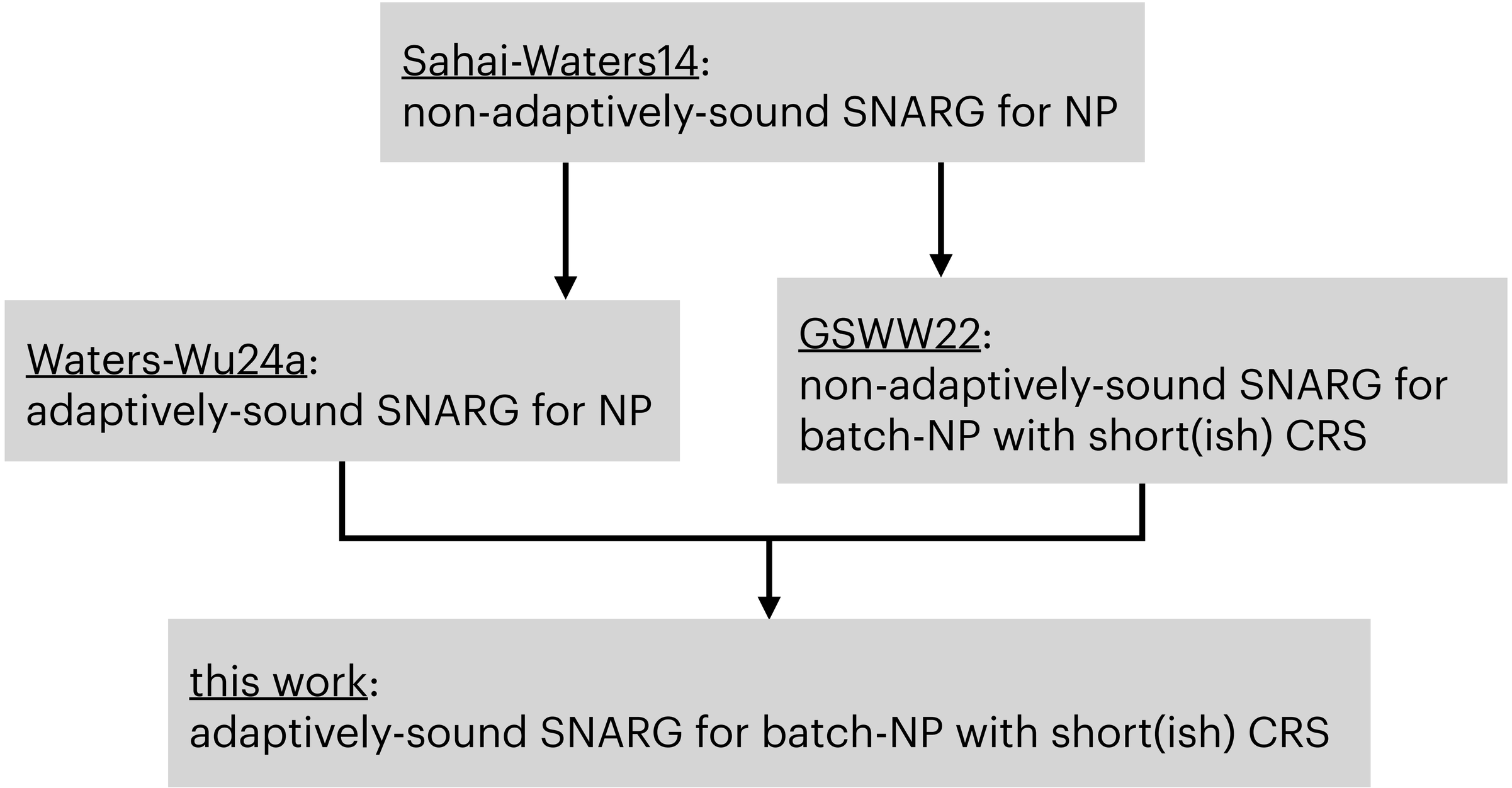
Sahai-Waters14:
non-adaptively-sound SNARG for NP

GSWW22:
non-adaptively-sound SNARG for batch-NP with short(ish) CRS

Sahai-Waters14:
non-adaptively-sound SNARG for NP

Waters-Wu24a:
adaptively-sound SNARG for NP

GSWW22:
non-adaptively-sound SNARG for batch-NP with short(ish) CRS


Sahai-Waters14:
non-adaptively-sound SNARG for NP
Waters-Wu24a:
adaptively-sound SNARG for NP
GSWW22:
non-adaptively-sound SNARG for batch-NP with short(ish) CRS
this work:
adaptively-sound SNARG for batch-NP with short(ish) CRS

Sahai-Waters template

Sahai-Waters template

- SNARG for statement x consists of a privately-verifiable signature (MAC)

Sahai-Waters template

- SNARG for statement x consists of a privately-verifiable signature (MAC)
- CRS contains two obfuscated programs with the MAC key k hardcoded inside:

Sahai-Waters template

- SNARG for statement x consists of a privately-verifiable signature (MAC)
- CRS contains two obfuscated programs with the MAC key k hardcoded inside:
  - Prove_k program signs the statement x if given a valid witness w

Sahai-Waters template

- SNARG for statement x consists of a privately-verifiable signature (MAC)
- CRS contains two obfuscated programs with the MAC key k hardcoded inside:
  - Prove_k program signs the statement x if given a valid witness w
  - Verify_k program checks that π is a valid signature for the statement x

Sahai-Waters template

- $C =$ NP verifier for L
- F is a puncturable PRF

- SNARG for statement x consists of a privately-verifiable signature (MAC)
- CRS contains two obfuscated programs with the MAC key k hardcoded inside:
  - Prove_k program signs the statement x if given a valid witness w
  - Verify_k program checks that π is a valid signature for the statement x

CRS (obfuscated programs)

$\text{Prove}_k(x, w)$:
If $C(x, w) = 1$, output $\pi = F_k(x)$.
Else, output $\perp$.

$\text{Verify}_k(x, \pi)$:
If $\pi = F_k(x)$, output 1.
Else, output 0.

Sahai-Waters template

- $C =$ NP verifier for L
- F is a puncturable PRF

- SNARG for statement x consists of a privately-verifiable signature (MAC)
- CRS contains two obfuscated programs with the MAC key k hardcoded inside:
  - Prove_k program signs the statement x if given a valid witness w
  - checks that π is a valid signature for the statement x

this is why the CRS size grows with $|C|$

CRS (obfuscated programs)

Prove$_k(x, w)$:
If $C(x, w) = 1$, output $\pi = F_k(x)$.
Else, output $\perp$.

Verify$_k(x, \pi)$:
If $\pi = F_k(x)$, output 1.
Else, output 0.

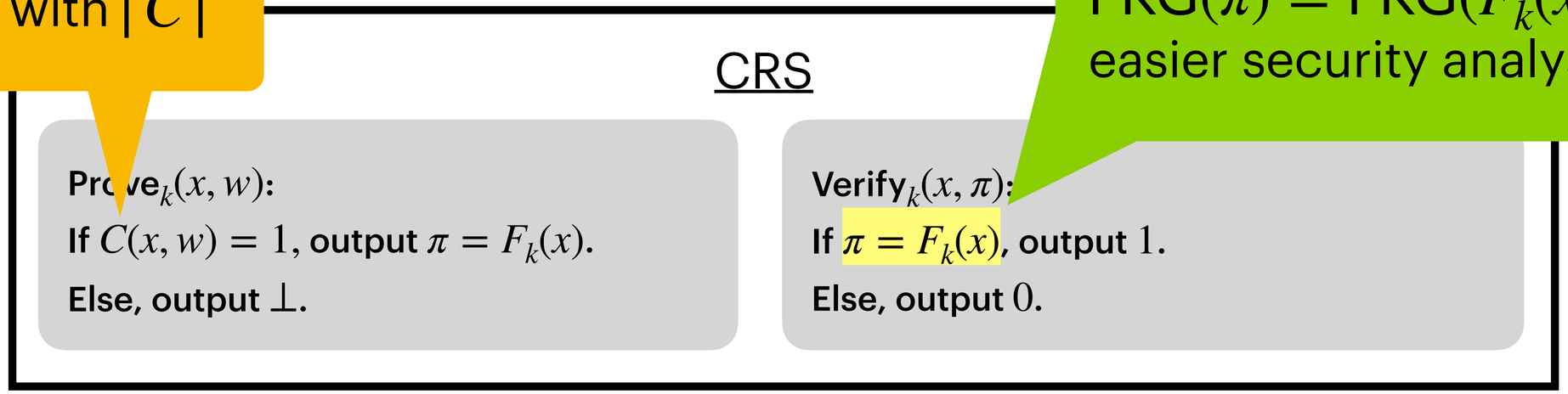
Sahai-Waters template

- C = NP verifier for L
- F is a puncturable PRF

- SNARG for statement x consists of a privately-verifiable signature (MAC)
- CRS contains two obfuscated programs with the MAC key k hardcoded inside:
  - Prove_k program signs the statement x if given a valid witness w
  - [illegible]n checks that π is a valid signature for

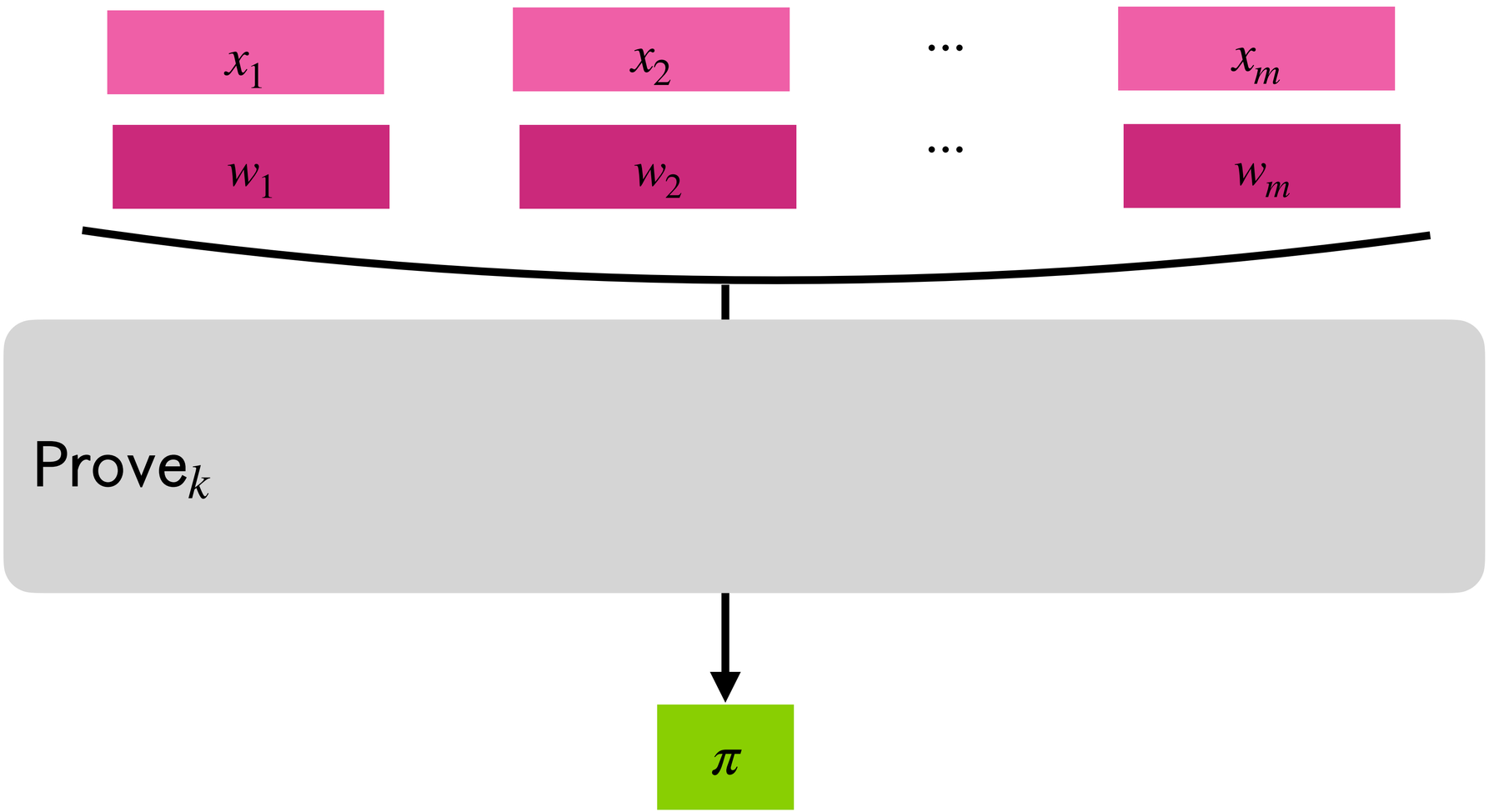
this is why the CRS size grows with $|C|$

will actually check $\text{PRG}(\pi) = \text{PRG}(F_k(x))$ for easier security analysis

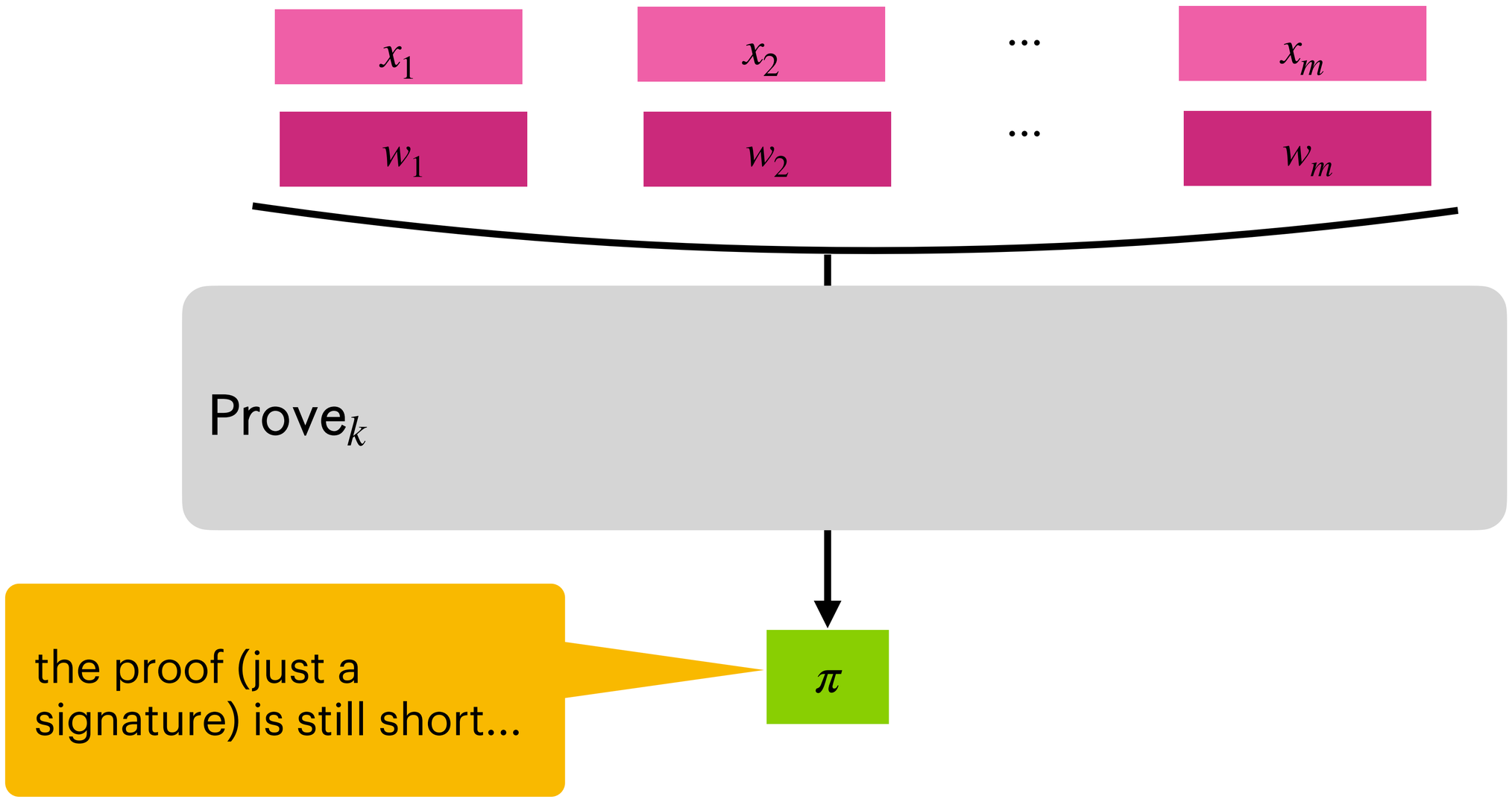


what if we use this template for batch-NP?

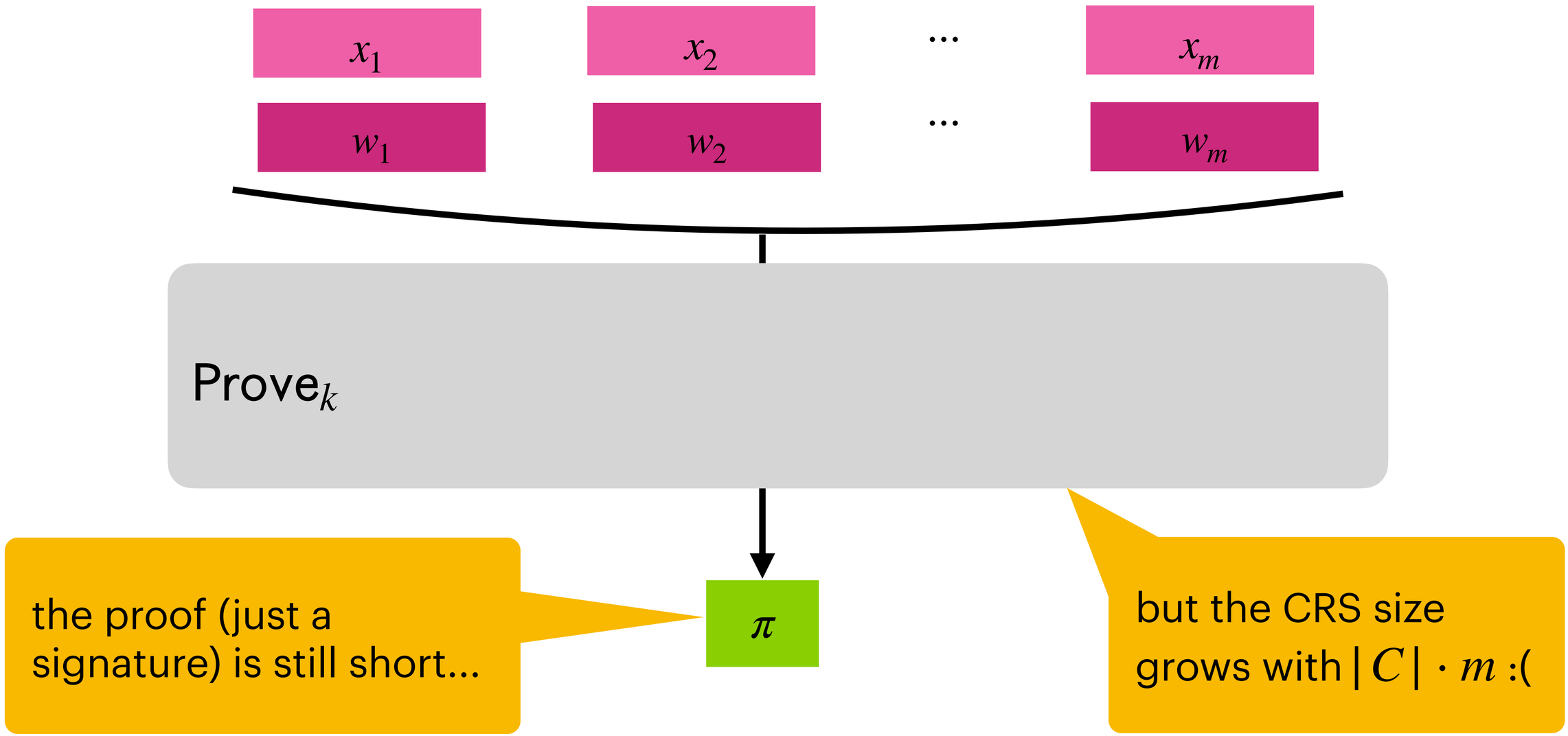
what if we use this template for batch-NP?



what if we use this template for batch-NP?



what if we use this template for batch-NP?



idea from GSWW22: batching via "chaining"

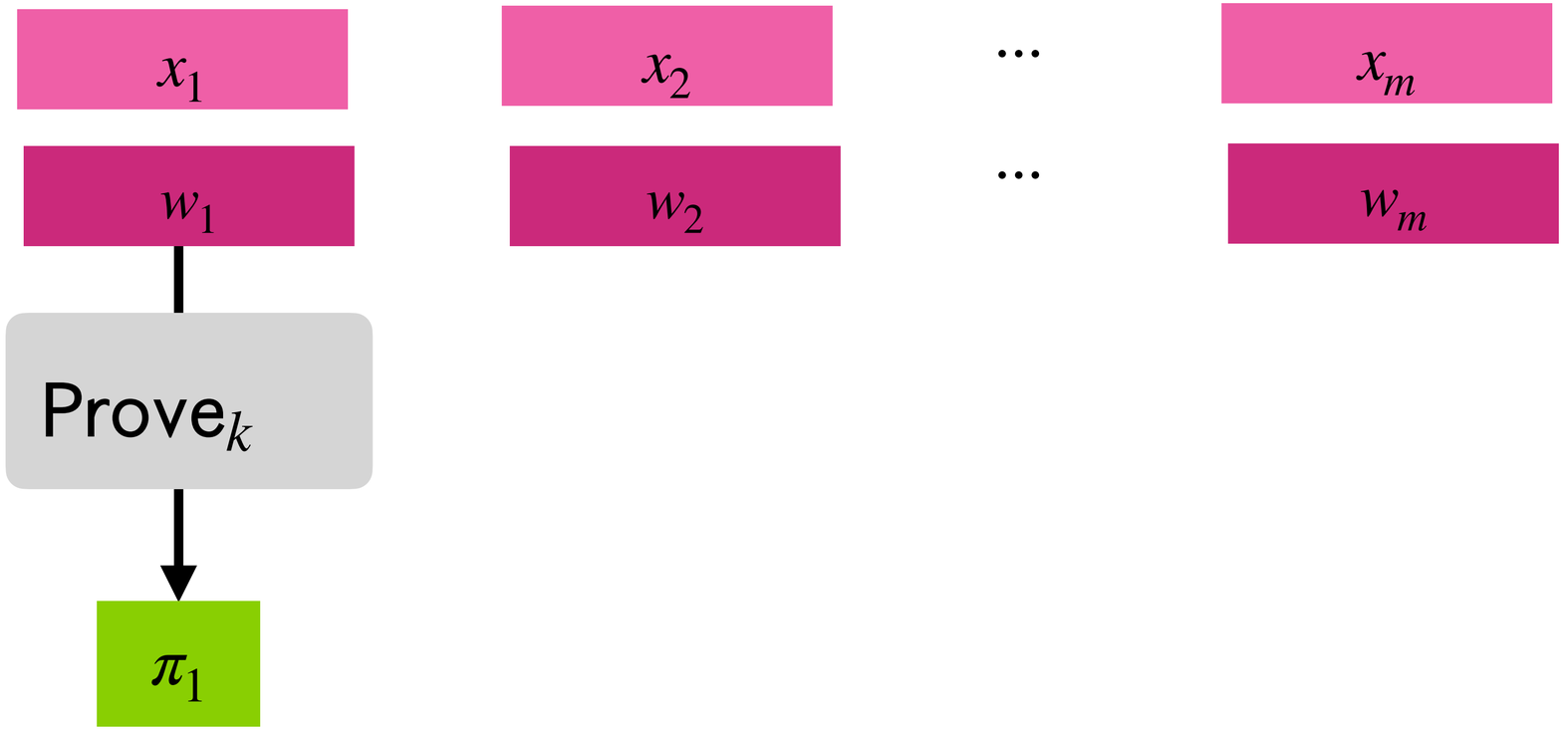
Prove$_k$ only outputs signature π_i if given

- a valid witness w_i
- a valid proof π_{i-1}

idea from GSWW22: batching via "chaining"

Prove_k only outputs signature π_i if given

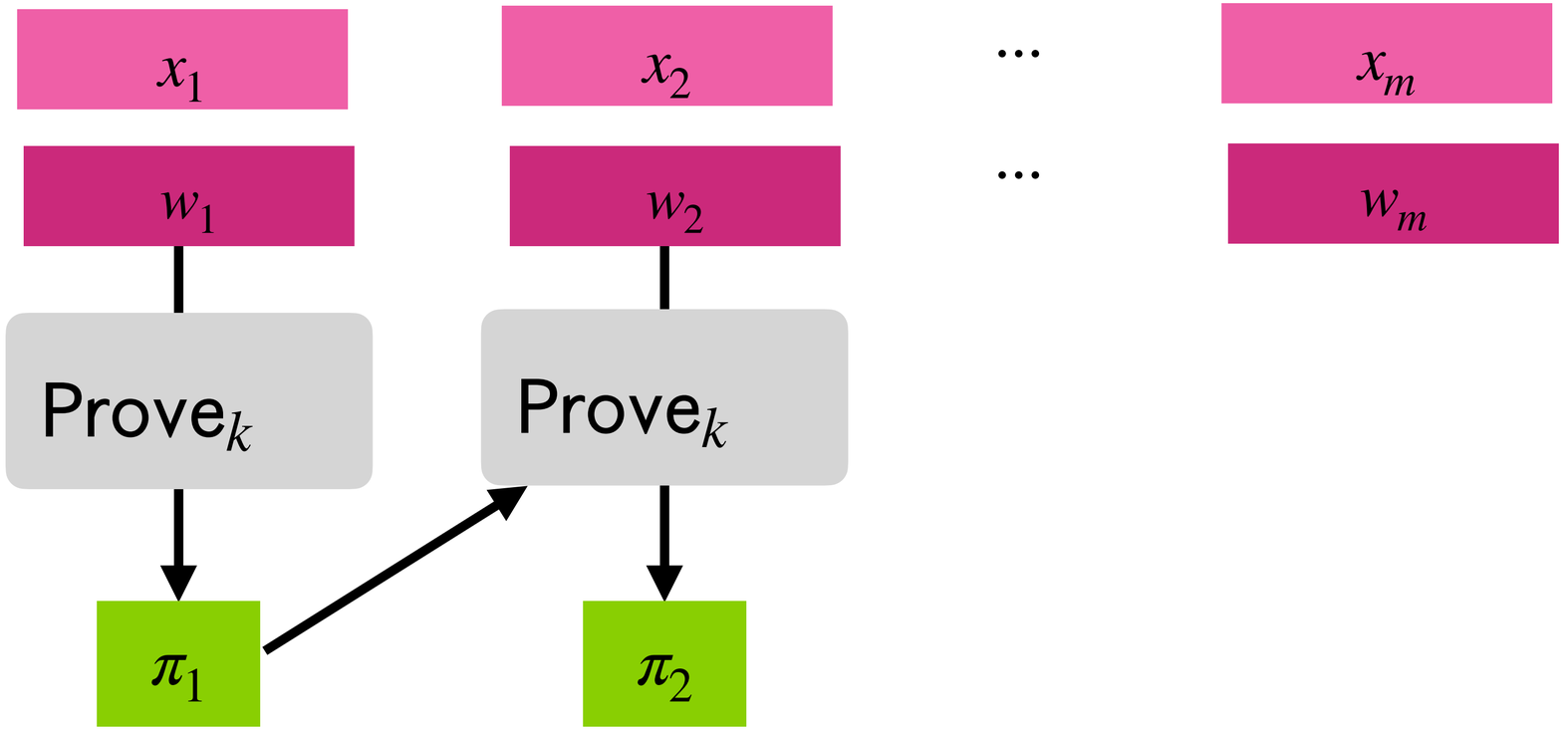
- a valid witness w_i
- a valid proof π_{i-1}



idea from GSWW22: batching via "chaining"

Prove_k only outputs signature π_i if given

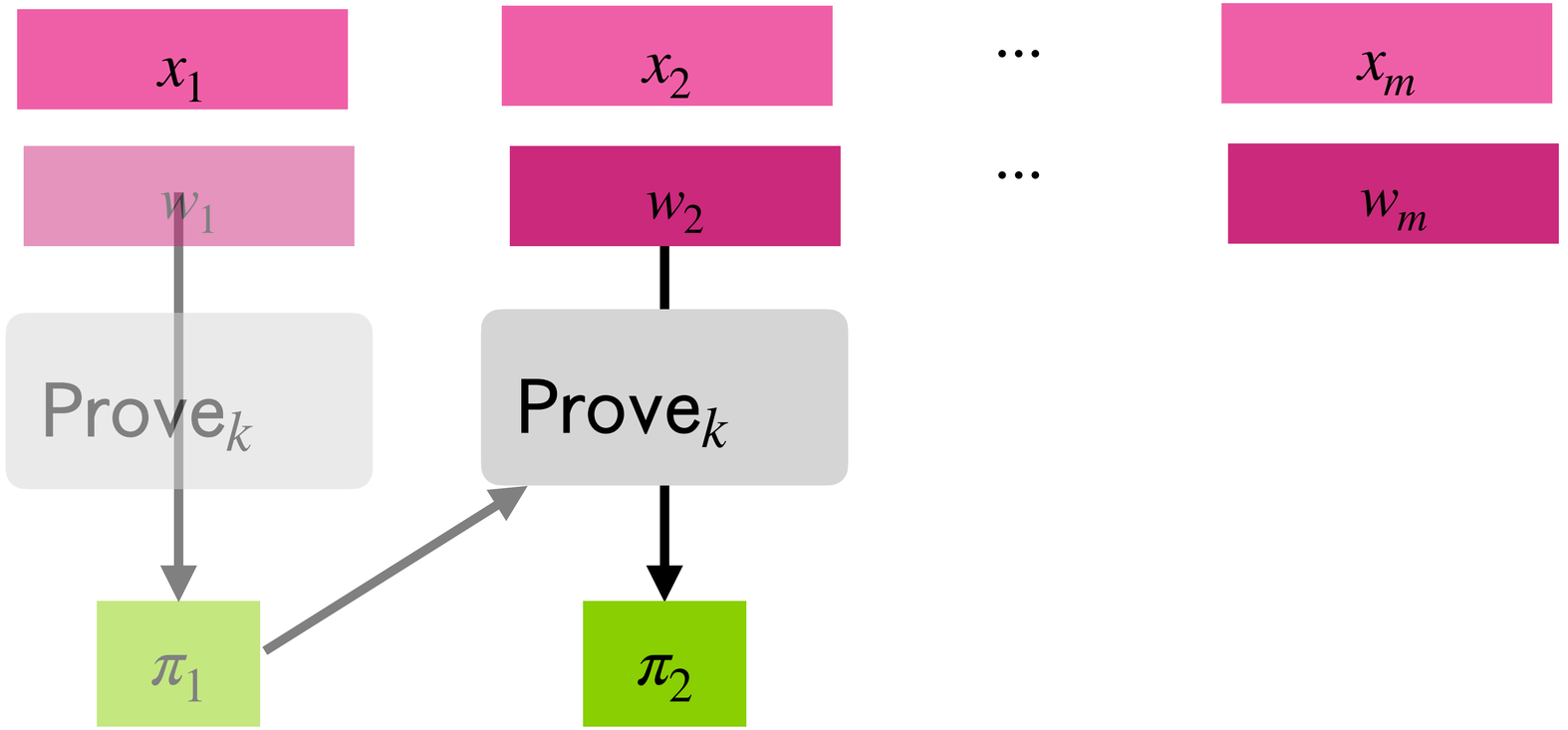
- a valid witness w_i
- a valid proof π_{i-1}



idea from GSWW22: batching via "chaining"

Prove_k only outputs signature π_i if given

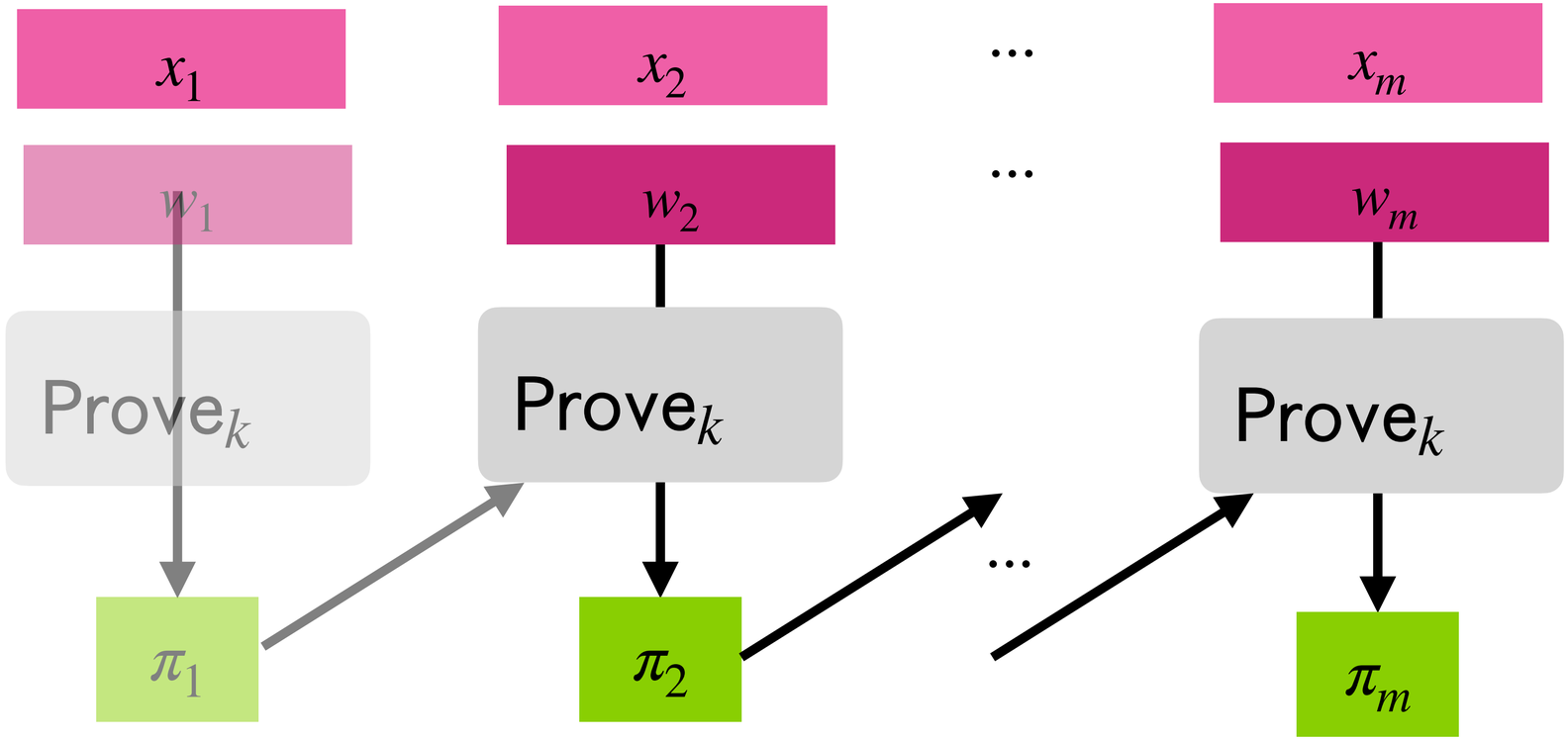
- a valid witness w_i
- a valid proof π_{i-1}



idea from GSWW22: batching via "chaining"

Prove_k only outputs signature π_i if given

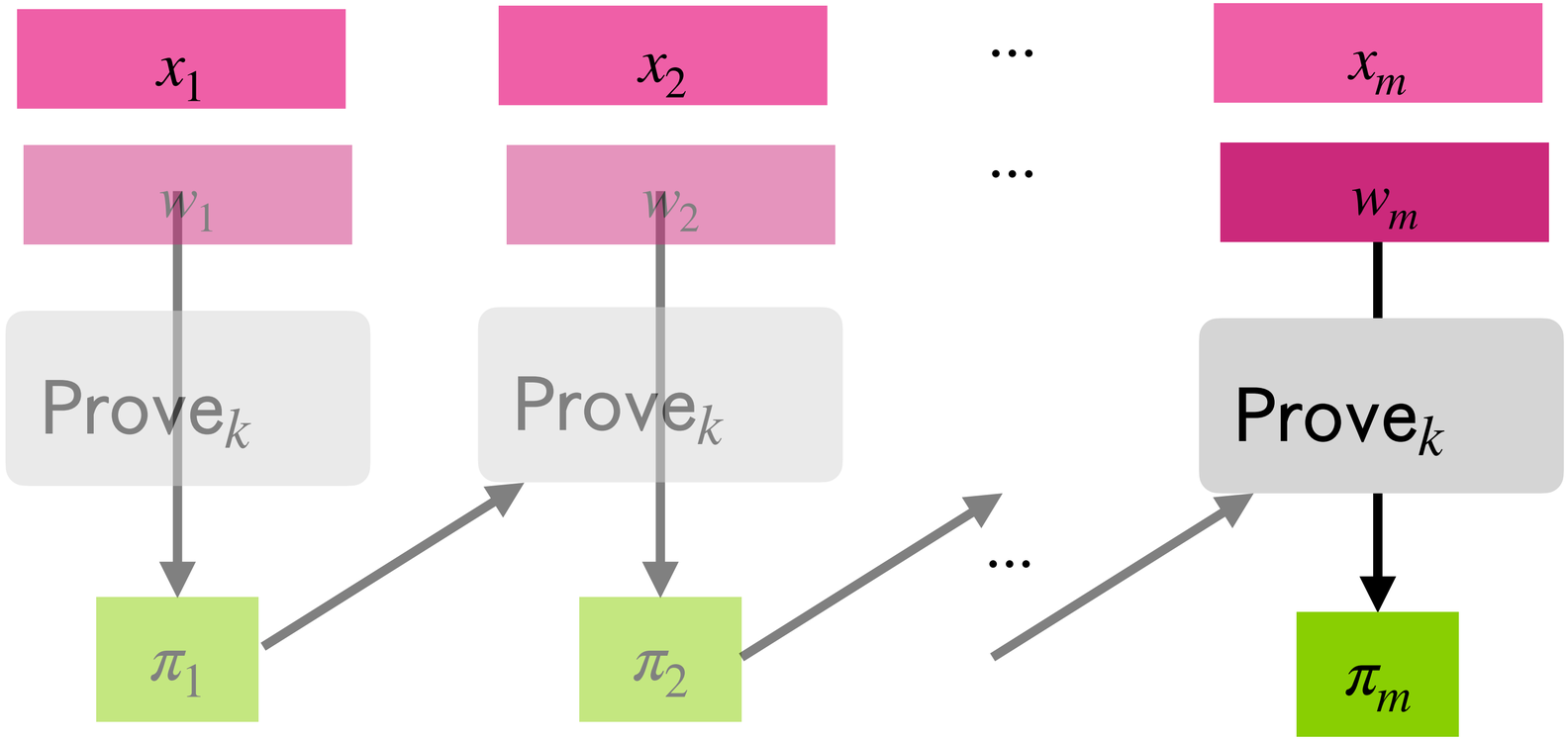
- a valid witness w_i
- a valid proof π_{i-1}



idea from GSWW22: batching via "chaining"

Prove_k only outputs signature π_i if given

- a valid witness w_i
- a valid proof π_{i-1}

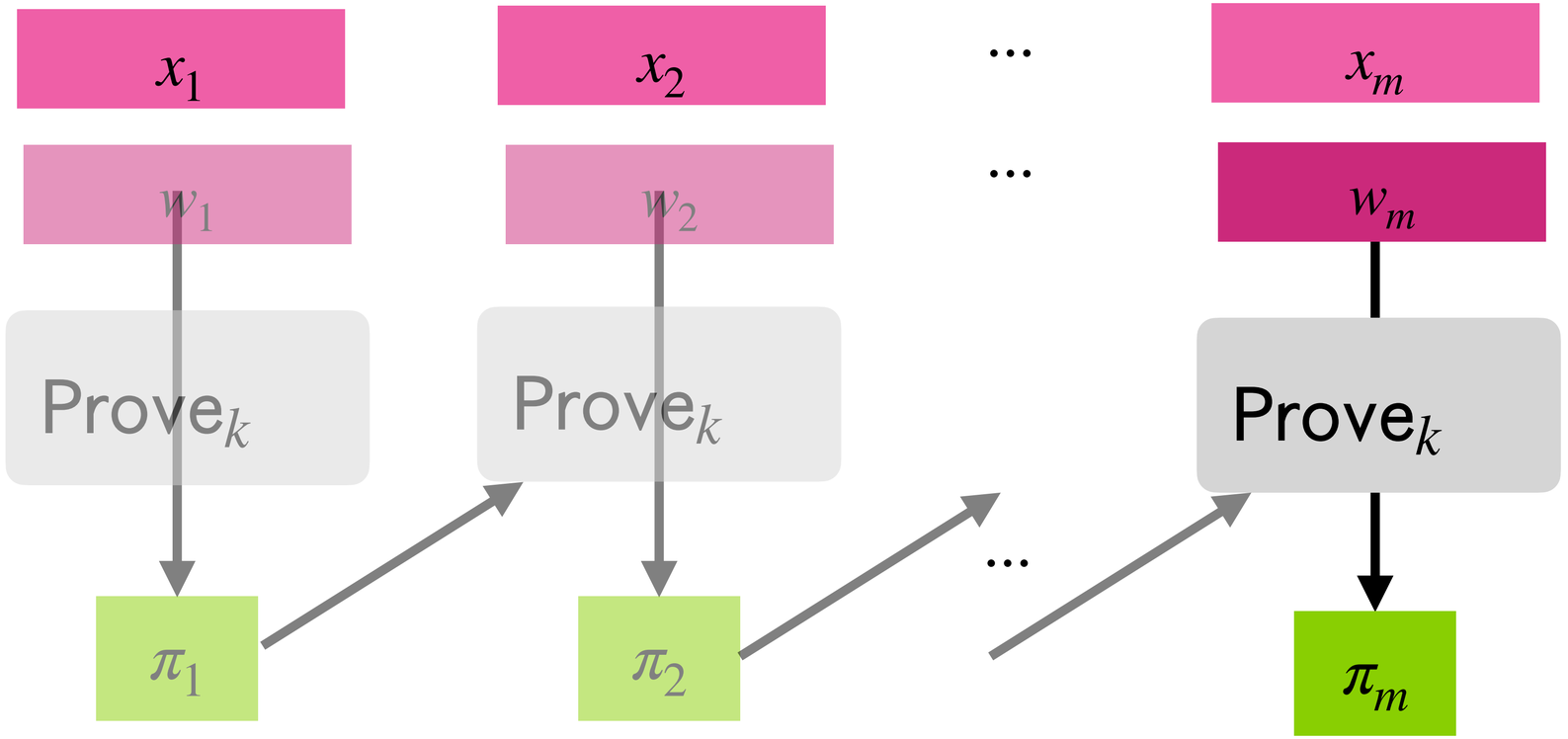


idea from GSWW22: batching via "chaining"

Prove_k only outputs signature π_i if given

- a valid witness w_i
- a valid proof π_{i-1}

Prove_k actually signs (h, i) where h is a hash of $(x_1, \ldots, x_m)$

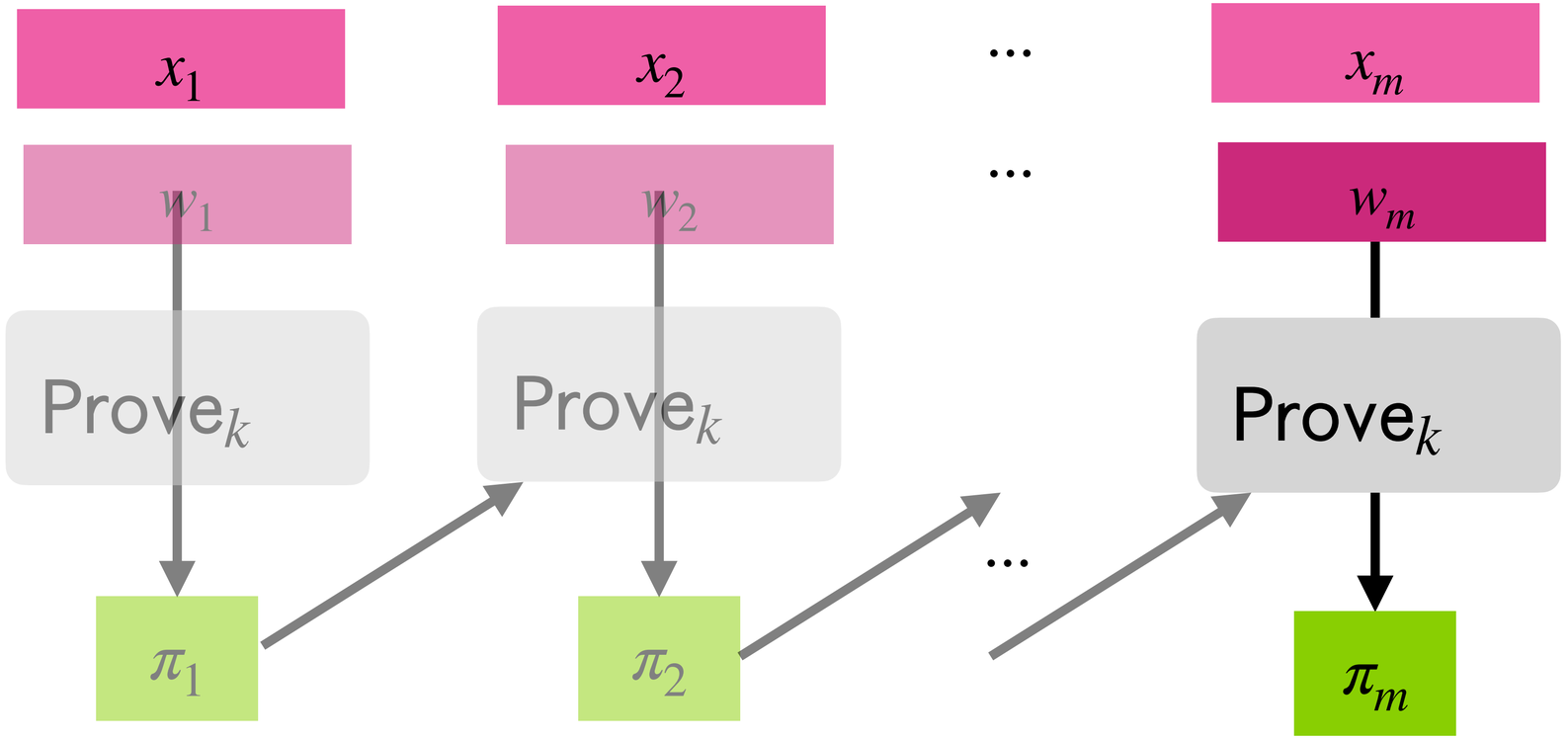


idea from GSWW22: batching via "chaining"

Prove_k only outputs signature π_i if given

- a valid witness w_i
- a valid proof π_{i-1}

Prove_k actually signs (h, i) where h is a hash of $(x_1, \ldots, x_m)$



so the obfuscated program in the CRS includes *one* copy of the NP verification circuit C (which the prover runs m times), rather than m copies!

achieving adaptive soundness

achieving adaptive soundness

- **problem**: we don't know the false statement x_{i*} in advance

achieving adaptive soundness

- **problem**: we don't know the false statement x_{i*} in advance
- **idea from WW24a**: use *exponentially many* hybrids to do an Sahai-Waters-like non-adaptive soundness analysis for every possible statement x_{i*} one-by-one

achieving adaptive soundness

- **problem**: we don't know the false statement x_{i*} in advance
- **idea from WW24a**: use *exponentially many* hybrids to do an Sahai-Waters-like non-adaptive soundness analysis for every possible statement x_{i*} one-by-one
- we need a **rerandomizable PRG** to make this work with the GSWW batching-via-chaining approach

DDH-based rerandomizable PRG

DDH-based rerandomizable PRG

$$\mathrm{PRG}(z) = (g^z, h^z)$$

DDH-based rerandomizable PRG

$$\text{PRG}(z) = (g^z, h^z)$$

- to re-randomize $y^* = (y_1, y_2)$, sample uniform r and output

DDH-based rerandomizable PRG

$$\text{PRG}(z) = (g^z, h^z)$$

- to re-randomize $y^* = (y_1, y_2)$, sample uniform r and output

$$\hat{y} = (y_1^r, y_2^r)$$

DDH-based rerandomizable PRG

$$\text{PRG}(z) = (g^z, h^z)$$

- to re-randomize $y^* = (y_1, y_2)$, sample uniform r and output

$$\hat{y} = (y_1^r, y_2^r)$$

perfectly uniform

DDH-based rerandomizable PRG

$$\text{PRG}(z) = (g^z, h^z)$$

- to re-randomize $y^* = (y_1, y_2)$, sample uniform r and output

$$\hat{y} = (y_1^r, y_2^r)$$

perfectly uniform

- if and only if $y^* = \text{PRG}(z) = (g^z, h^z)$ for some z, then

DDH-based rerandomizable PRG

$$\text{PRG}(z) = (g^z, h^z)$$

- to re-randomize $y^* = (y_1, y_2)$, sample uniform r and output

$$\hat{y} = (y_1^r, y_2^r)$$

perfectly uniform

- if and only if $y^* = \text{PRG}(z) = (g^z, h^z)$ for some z, then

$$\hat{y} = (g^{zr}, h^{zr}) = \text{PRG}(zr)$$

this work

theorem. there exists an adaptively-sound SNARG for batch-NP with CRS size poly$(\lambda, |C|, \log m)$ **and proof size** poly(λ).

- assuming iO, OWF, and DDH

future directions

future directions

- can we get a short(ish) CRS, i.e., not growing with the statement length, for other languages besides batch-NP?

future directions

- can we get a short(ish) CRS, i.e., not growing with the statement length, for other languages besides batch-NP?
  - threshold or monotone-policy batch-NP?

future directions

- can we get a short(ish) CRS, i.e., not growing with the statement length, for other languages besides batch-NP?
  - threshold or monotone-policy batch-NP?
- can we build an adaptively-sound SNARG from only subexponentially-secure iO+OWF like in WW24b?

future directions

- can we get a short(ish) CRS, i.e., not growing with the statement length, for other languages besides batch-NP?
  - threshold or monotone-policy batch-NP?
- can we build an adaptively-sound SNARG from only subexponentially-secure iO+OWF like in WW24b?
- can we construct rerandomizable PRGs or homomorphic rerandomizable OWFs from other assumptions?

thank you!

full version: eprint.iacr.org/2024/1812

questions?