Batch Arguments for NP from Standard Bilinear Group Assumptions

Brent Waters and David Wu

Batch Arguments for NP

Boolean circuit satisfiability

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

prover

$(x_1, \dots, x_m)$

prover has m statements and wants to convince verifier that $x_i \in \mathcal{L}_C$ for all $i \in [m]$

verifier

Batch Arguments for NP

Boolean circuit satisfiability

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

prover

$(x_1, \dots, x_m)$

$\pi = (w_1, \dots, w_m)$

verifier

Naïve solution: send witnesses $w_1, \dots, w_m$ and verifier checks $C(x_i, w_i) = 1$ for all $i \in [m]$

Can the proof size be sublinear in the number of instances m?

Goal: Amortize the Cost of NP Verification

Boolean circuit satisfiability

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

prover

$(x_1, \dots, x_m)$

$\longrightarrow$

π

$\longrightarrow$

verifier

Proof size: $|\pi| = \mathrm{poly}(\lambda, \log m, |C|)$

λ : security parameter

Proof size can scale with circuit size (**not** a SNARG for NP)

Goal: Amortize the Cost of NP Verification

Boolean circuit satisfiability

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

prover

$$(x_1, \ldots, x_m) \longrightarrow$$

$$\pi \longrightarrow$$

verifier

Proof size: $|\pi| = \text{poly}(\lambda, \log m, |C|)$

In general setting, verifier needs to read statements

Verification time: running time of verifier is $\text{poly}(\lambda, m, n) + \text{poly}(\lambda, \log m, |C|)$

Goal: Amortize the Cost of NP Verification

Boolean circuit satisfiability

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

prover

$$\xrightarrow{(x_1, \dots, x_m)}$$

$$\xrightarrow{\pi}$$

verifier

Computational soundness: polynomial-time prover cannot convince verifier of $(x_1, \dots, x_m)$ if there is any $i \in [m]$ where $x_i \notin \mathcal{L}_C$

Goal: Amortize the Cost of NP Verification

Boolean circuit satisfiability

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

For (statistically-sound) proofs:

- With inefficient provers, IP = PSPACE [LFKN92, Sha92] theorem gives interactive proof for batch NP with communication $\text{poly}(\log m, |C|)$
- With efficient provers, we have interactive proofs for batch UP with communication $\text{poly}(\log m, |C|)$ [RRR16, RRR18, RR20]

verifier

Computational soundness: polynomial-time prover cannot convince verifier of $(x_1, \ldots, x_m)$ if there is any $i \in [m]$ where $x_i \notin \mathcal{L}_C$

Goal: Amortize the Cost of NP Verification

Boolean circuit satisfiability

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

prover 

$$\xrightarrow{(x_1, \dots, x_m)}$$

$$\xrightarrow{\pi}$$

 verifier

Focus: Non-interactive setting (proof is a single message)

Goal: Amortize the Cost of NP Verification

common reference string

110010000111111000001101111101111111011111010010010

prover 

$(x_1, \dots, x_m)$ $\longrightarrow$

π $\longrightarrow$

 verifier

Focus: Non-interactive setting (proof is a single message)

Prover and verifier have access to a common reference string (CRS)

An Application: Succinct Argument for P

[KPY19, CJJ21]

Turing machine M, input x, time bound T

Show: $M(x) = 1$ in at most T steps

prover

$$(x, T)$$

$$\pi$$

verifier

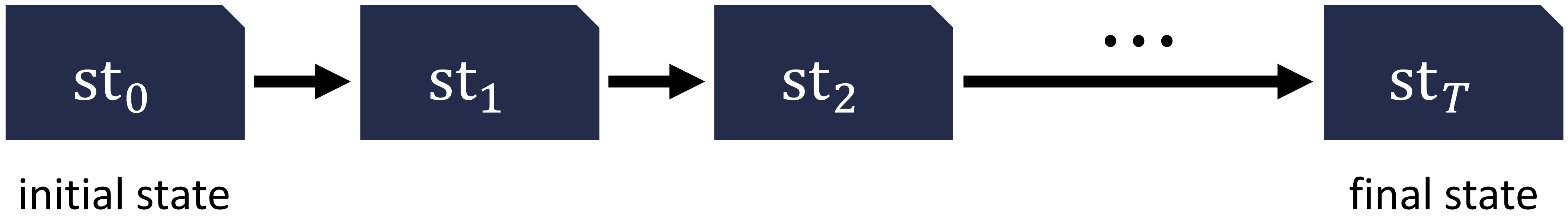
Proof size: $|\pi| = \mathrm{poly}(\lambda, \log T)$

Verification time: running time of verifier is $\mathrm{poly}(\lambda, |x|) + \mathrm{poly}(\lambda, \log T)$

An Application: Succinct Argument for P

[KPY19, CJJ21]

(Very) high-level idea:



Prover commits to the vector of computation states $(\mathrm{st}_0, \ldots, \mathrm{st}_T)$

Checking each transition can be implemented by a circuit of size $\mathrm{poly}(\lambda)$

Each step only changes a constant number of positions in the computation state

Prover constructs a batch argument that all T transitions are valid

Statements are indices $1, \ldots, T$ *and the NP relation is checking validity of step* i

Batch Arguments for NP

Special case of succinct non-interactive arguments for NP (SNARGs)

Constructions rely on idealized models or knowledge assumptions or indistinguishability obfuscation

Batch arguments from correlation intractable hash functions

Sub-exponential DDH (in pairing-free groups) + QR (with $\sqrt{m}$ size proofs) [CJJ21a]

Learning with errors (LWE) [CJJ21b]

Batch arguments from pairing-based assumptions

Non-standard, but falsifiable q-type assumption on bilinear groups [KPY19]

This Work

New constructions of non-interactive batch arguments for NP

Batch arguments for NP from standard assumptions over bilinear maps

k-Linear assumption (for any $k \geq 1$) in prime-order bilinear groups

Subgroup decision assumption in composite-order bilinear groups

Key feature: Construction is "low-tech"

No heavy tools like correlation-intractable hash functions or probabilistically-checkable proofs

Direct "commit-and-prove" approach à la classic NIZK construction of Groth-Ostrovsky-Sahai

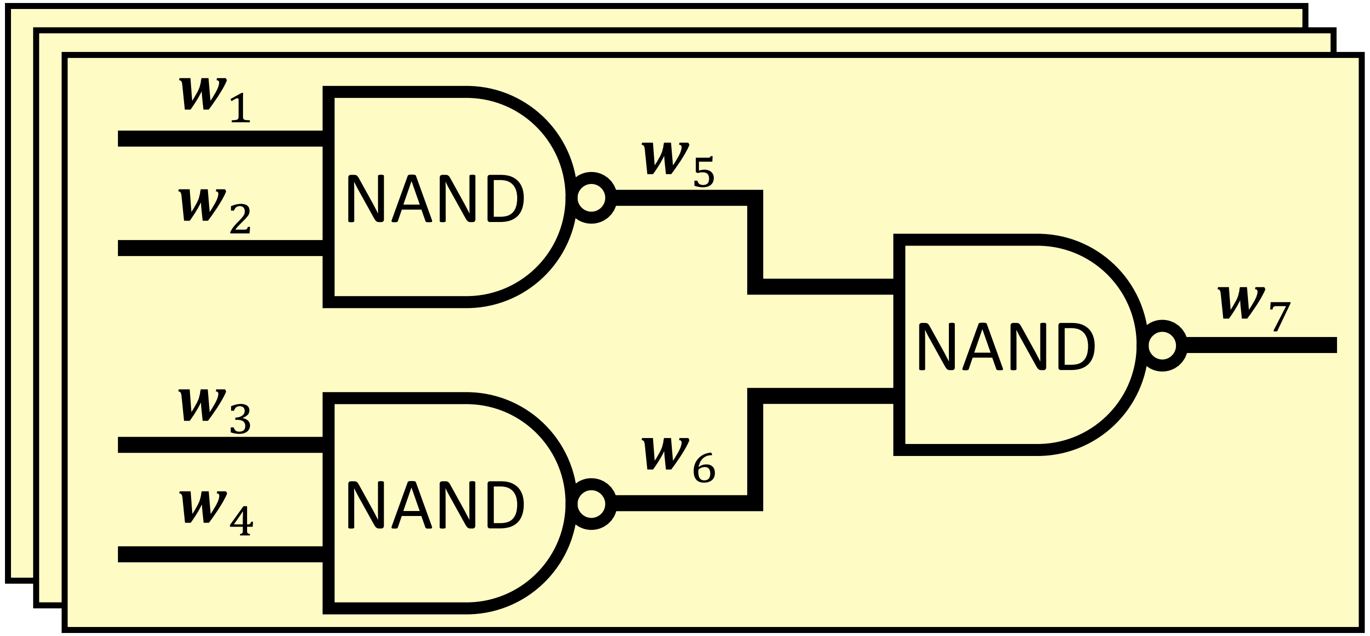
Corollary: RAM delegation (i.e., "SNARG for P") with sublinear CRS from standard bilinear map assumptions

Previous bilinear map constructions: need non-standard assumptions [KPY19] or have long CRS [GZ21]

Corollary: Aggregate signature with bounded aggregation from standard bilinear map assumptions

Previous bilinear map constructions: random oracle based [BGLS03]

A Commit-and-Prove Strategy for Batch Arguments



Let $\boldsymbol{w}_i = (w_{i,1}, \ldots, w_{i,m})$ be vector of wire labels associated with wire i across the m instances

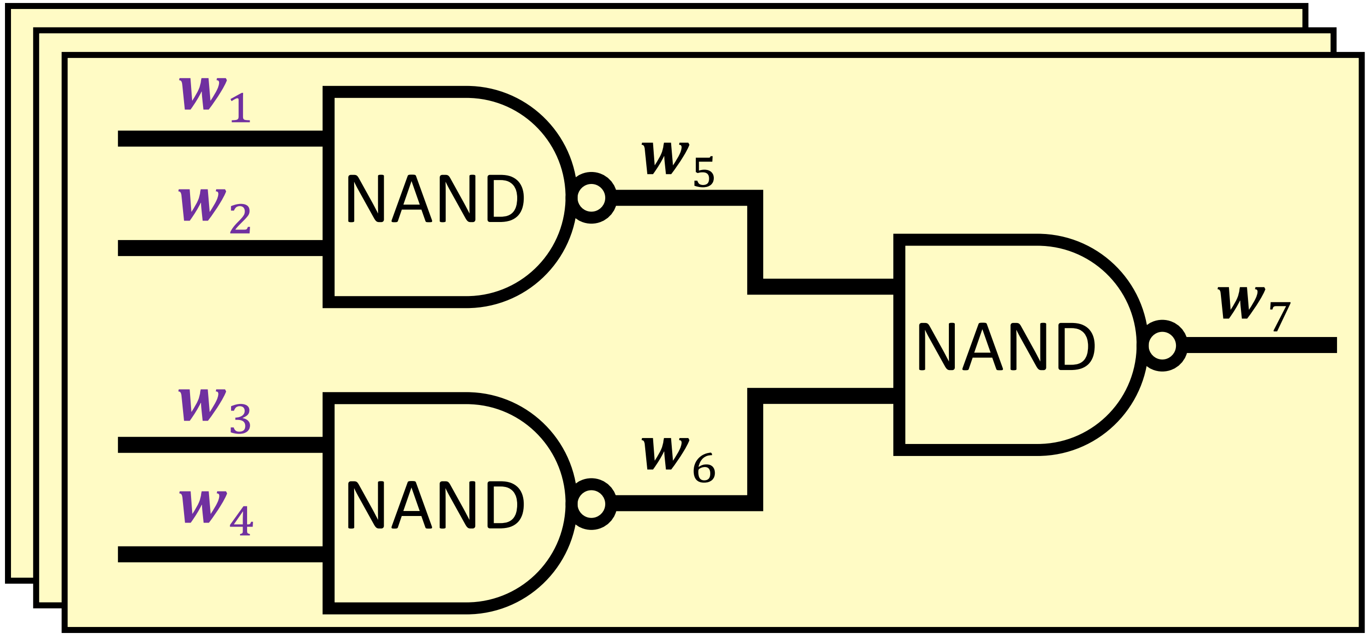
1. Prover commits to each vector of wire assignments

$$\boldsymbol{w}_i = \boxed{w_{i,1}}\ \boxed{w_{i,2}}\ \boxed{\cdots}\ \boxed{w_{i,m}} \longrightarrow \boxed{\sigma_i}$$

Requirement: $|\sigma_i| = \mathrm{poly}(\lambda, \log m)$

Our construction: $|\sigma_i| = \mathrm{poly}(\lambda)$

A Commit-and-Prove Strategy for Batch Arguments



Let $\boldsymbol{w}_i = (w_{i,1}, \dots, w_{i,m})$ be vector of wire labels associated with wire i across the m instances

1 Prover commits to each vector of wire assignments

$$\boldsymbol{w}_i = \boxed{w_{i,1}}\ \boxed{w_{i,2}}\ \boxed{\cdots}\ \boxed{w_{i,m}} \longrightarrow \boxed{\sigma_i}$$

Requirement: $|\sigma_i| = \mathrm{poly}(\lambda, \log m)$

Our construction: $|\sigma_i| = \mathrm{poly}(\lambda)$

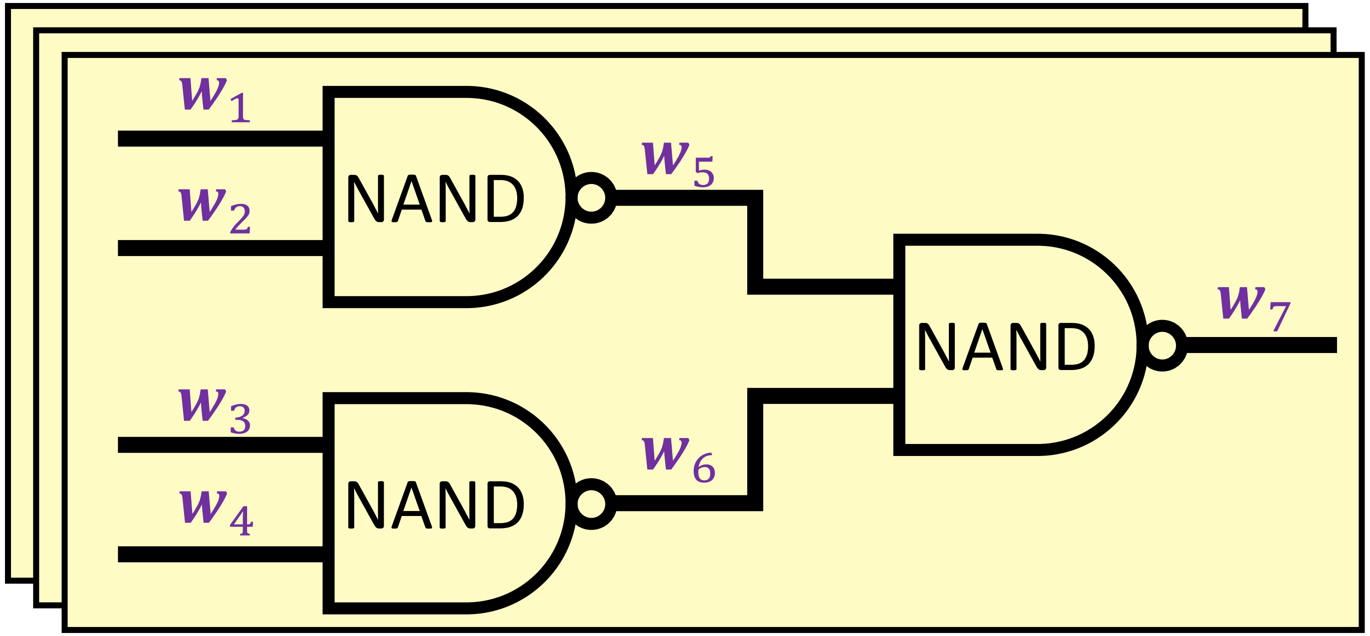
2 Prover constructs the following proofs:

Input validity

Commitments to the statement wires are correctly computed

Commitments in our scheme are *deterministic*, so verifier can directly check

A Commit-and-Prove Strategy for Batch Arguments



Let $\boldsymbol{w}_i = (w_{i,1}, \ldots, w_{i,m})$ be vector of wire labels associated with wire i across the m instances

1 Prover commits to each vector of wire assignments

$$\boldsymbol{w}_i = \boxed{w_{i,1}}\ \boxed{w_{i,2}}\ \boxed{\cdots}\ \boxed{w_{i,m}} \longrightarrow \boxed{\sigma_i}$$

Requirement: $|\sigma_i| = \mathrm{poly}(\lambda, \log m)$

Our construction: $|\sigma_i| = \mathrm{poly}(\lambda)$

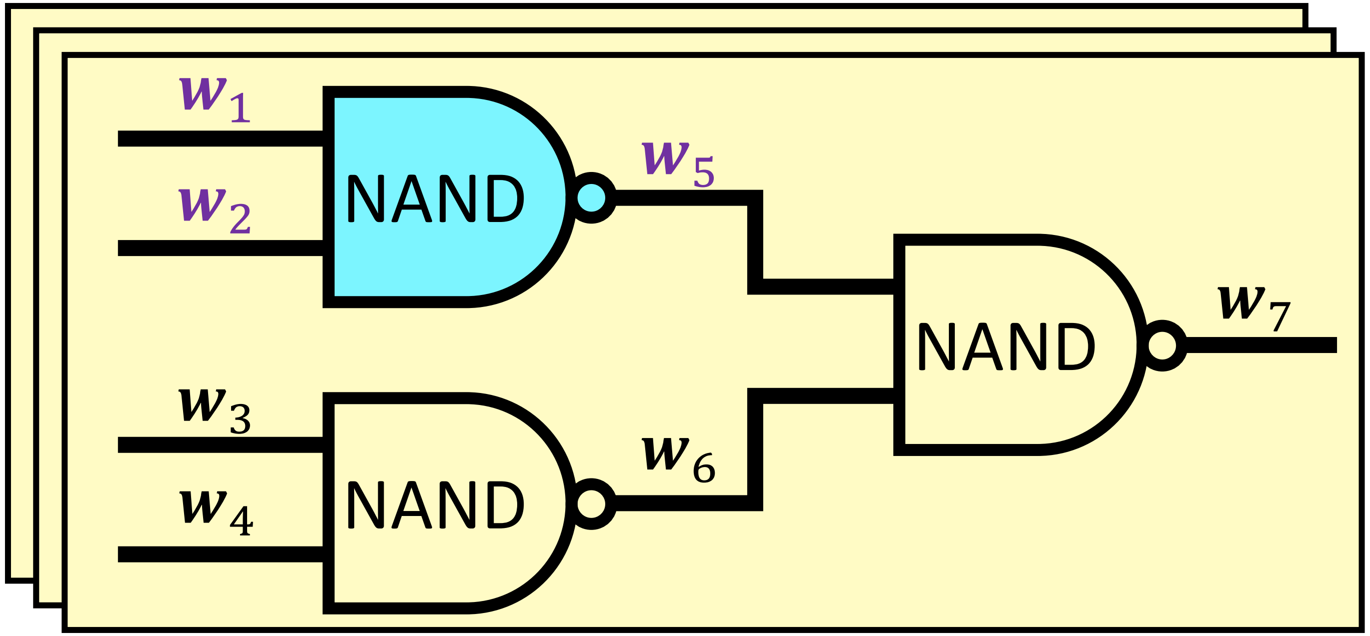
2 Prover constructs the following proofs:

Input validity

Wire validity

Commitment for each wire is a commitment to a 0/1 vector

A Commit-and-Prove Strategy for Batch Arguments



Let $\boldsymbol{w}_i = (w_{i,1}, \dots, w_{i,m})$ be vector of wire labels associated with wire i across the m instances

1 Prover commits to each vector of wire assignments

$$\boldsymbol{w}_i = \boxed{w_{i,1}}\ \boxed{w_{i,2}}\ \boxed{\cdots}\ \boxed{w_{i,m}} \longrightarrow \boxed{\sigma_i}$$

Requirement: $|\sigma_i| = \text{poly}(\lambda, \log m)$

Our construction: $|\sigma_i| = \text{poly}(\lambda)$

2 Prover constructs the following proofs:

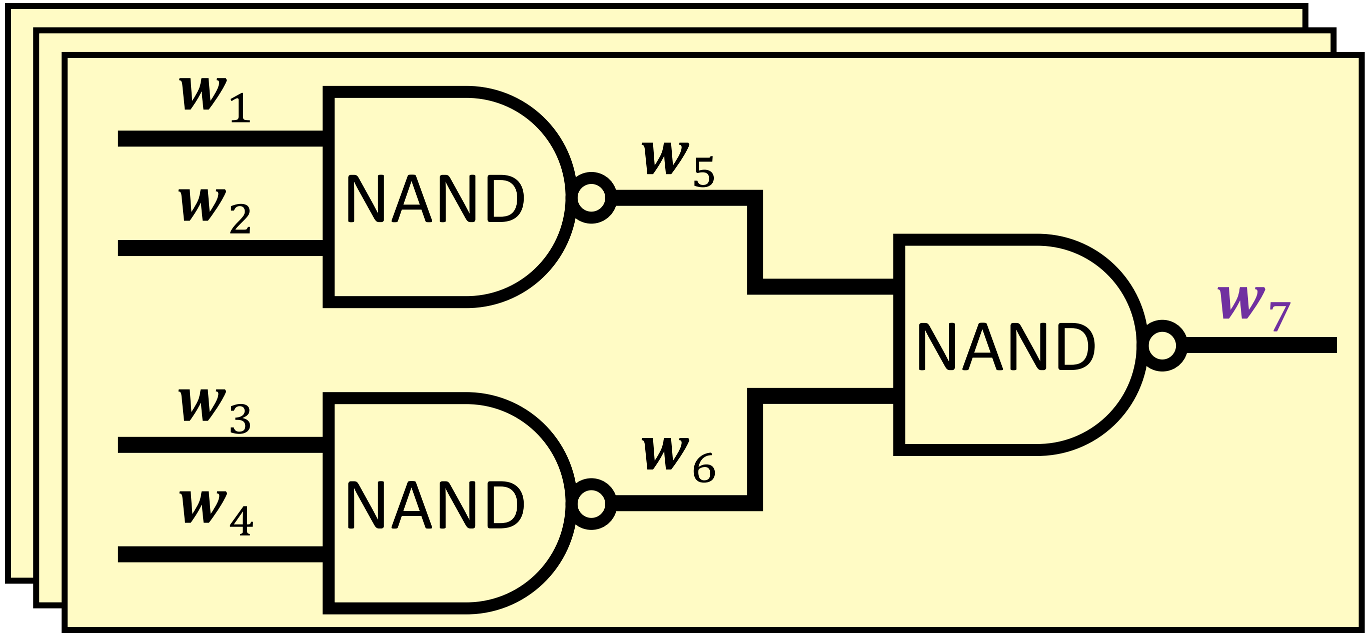
Input validity

Wire validity

Gate validity

For each gate, commitment to output wires is consistent with gate operation and commitment to input wires

A Commit-and-Prove Strategy for Batch Arguments



Let $\boldsymbol{w}_i = (w_{i,1}, \ldots, w_{i,m})$ be vector of wire labels associated with wire i across the m instances

1 Prover commits to each vector of wire assignments

$$\boldsymbol{w}_i = \boxed{w_{i,1}}\ \boxed{w_{i,2}}\ \boxed{\cdots}\ \boxed{w_{i,m}} \Rightarrow \boxed{\sigma_i}$$

Requirement: $|\sigma_i| = \mathrm{poly}(\lambda, \log m)$

Our construction: $|\sigma_i| = \mathrm{poly}(\lambda)$

2 Prover constructs the following proofs:

Input validity

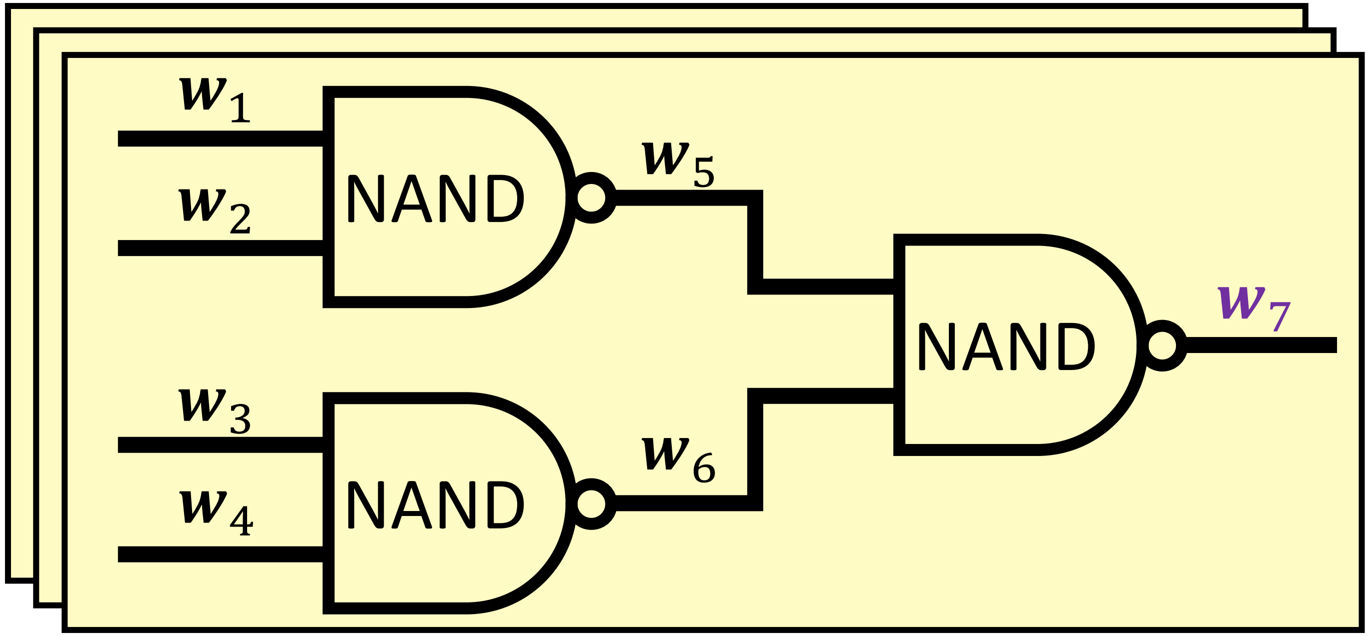
Wire validity

Gate validity

Output validity

Commitment to output wire is a commitment to the all-ones vector

A Commit-and-Prove Strategy for Batch Arguments



Let $\boldsymbol{w}_i = (w_{i,1}, \dots, w_{i,m})$ be vector of wire labels associated with wire i across the m instances

1 Prover commits to each vector of wire assignments

$\boldsymbol{w}_i =$ $w_{i,1}$ $w_{i,2}$ $\cdots$ $w_{i,m}$ → σ_i

Requirement: $|\sigma_i| = \mathrm{poly}(\lambda, \log m)$

Our construction: $|\sigma_i| = \mathrm{poly}(\lambda)$

2 Prover constructs the following proofs:

- **Input validity**
- **Wire validity**
- **Gate validity**
- **Output validity**

Key idea: Validity checks are quadratic and can be checked in the exponent

Construction from Composite-Order Groups

Pedersen multi-commitments: (*without* randomness)

Let $\mathbb{G}$ be a group of order $N = pq$ (composite order)

Let $\mathbb{G}_p \subset \mathbb{G}$ be the subgroup of order p and let g_p be a generator of $\mathbb{G}_p$

crs: sample $\alpha_1, \ldots, \alpha_m \leftarrow \mathbb{Z}_N$

output $A_1 \leftarrow g_p^{\alpha_1}, \ldots, A_m \leftarrow g_p^{\alpha_m}$

denotes encodings in $\mathbb{G}_p$

$\boxed{[\alpha_1]}\ \boxed{[\alpha_2]}\ \boxed{[\cdots]}\ \boxed{[\alpha_m]}$

commitment to $\boldsymbol{x} = (x_1, \ldots, x_m) \in \{0,1\}^m$:

$$\sigma_{\boldsymbol{x}} = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m}$$

(subset product of the A_i's)

$$\boxed{[\sigma_{\boldsymbol{x}}]} = \boxed{\left[\Sigma_{i \in [m]} \alpha_i x_i\right]}$$

Proving Relations on Committed Values

Common reference string:

$[\alpha_1]$ $A_1 = g_p^{\alpha_1}$

$[\vdots]$

$[\alpha_m]$ $A_m = g_p^{\alpha_m}$

Commitment to $(x_1, \dots, x_m)$:

$[\Sigma_{i \in [m]} \alpha_i x_i]$

$$\sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m} = g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}$$

Wire validity

Commitment for each wire is a commitment to a 0/1 vector

$x \in \{0,1\}$ if and only if $x^2 = x$

Key idea: Use pairing to check quadratic relation in the exponent

Recall: pairing is an efficiently-computable bilinear map on $\mathbb{G}$:

$$e(g^x, g^y) = e(g,g)^{xy}$$

$$e([x], [y]) \longrightarrow [xy]$$

Multiplies exponents in the target group

Proving Relations on Committed Values

Common reference string:

$[\alpha_1]$ $A_1 = g_p^{\alpha_1}$

$[\vdots]$

$[\alpha_m]$ $A_m = g_p^{\alpha_m}$

Commitment to $(x_1, \dots, x_m)$:

$[\Sigma_{i\in[m]} \alpha_i x_i]$

$$\sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m}$$
$$= g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}$$

Wire validity

Commitment for each wire is a commitment to a 0/1 vector

$x \in \{0,1\}$ if and only if $x^2 = x$

Approach: consider the following pairing relations:

$e(\sigma_x, \sigma_x)$ and $e\left(\sigma_x, \Pi_{i\in[m]} A_i\right)$

$$A = \Pi_{i\in[m]} A_i = g_p^{\Sigma_{i\in[m]} \alpha_i}$$

(commitment to all-ones vector)

Proving Relations on Committed Values

Common reference string:

$[\alpha_1]$ $A_1 = g_p^{\alpha_1}$

$[\vdots]$

$[\alpha_m]$ $A_m = g_p^{\alpha_m}$

Commitment to $(x_1, \dots, x_m)$:

$[\Sigma_{i\in[m]}\alpha_i x_i]$

$$\sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m}$$
$$= g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}$$

Wire validity

Commitment for each wire is a commitment to a 0/1 vector

$x \in \{0,1\}$ if and only if $x^2 = x$

Approach: consider the following pairing relations:

$e(\sigma_x, \sigma_x)$ and $e\left(\sigma_x, \Pi_{i\in[m]} A_i\right)$

$$e\left([\Sigma_{i\in[m]}\alpha_i x_i], [\Sigma_{i\in[m]}\alpha_i x_i]\right)$$

$$= \underbrace{[\Sigma_{i\in[m]}\alpha_i^2 x_i^2]}_{\textit{non-cross terms}} \times \underbrace{[\Sigma_{i\neq j}\alpha_i\alpha_j x_i x_j]}_{\textit{cross terms}}$$

Proving Relations on Committed Values

Common reference string:

$[\alpha_1]$ $A_1 = g_p^{\alpha_1}$

$[\vdots]$

$[\alpha_m]$ $A_m = g_p^{\alpha_m}$

Wire validity

Commitment for each wire is a commitment to a 0/1 vector

$x \in \{0,1\}$ if and only if $x^2 = x$

Approach: consider the following pairing relations:

$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}})$ and $e\left(\sigma_{\boldsymbol{x}}, \Pi_{i\in[m]} A_i\right)$

$$e\left([\Sigma_{i\in[m]}\alpha_i x_i], [\Sigma_{i\in[m]}\alpha_i]\right) = [\Sigma_{i\in[m]}\alpha_i^2 x_i] \times [\Sigma_{i\neq j}\alpha_i \alpha_j x_i]$$

non-cross terms *cross terms*

$$e\left([\Sigma_{i\in[m]}\alpha_i x_i], [\Sigma_{i\in[m]}\alpha_i x_i]\right) = [\Sigma_{i\in[m]}\alpha_i^2 x_i^2] \times [\Sigma_{i\neq j}\alpha_i \alpha_j x_i x_j]$$

non-cross terms *cross terms*

Proving Relations on Committed Values

If $x_i^2 = x_i$ for all i, then

$$[\Sigma_{i\in[m]}\alpha_i^2 x_i] = [\Sigma_{i\in[m]}\alpha_i^2 x_i^2]$$

Wire validity

Commitment for each wire is a commitment to a 0/1 vector

$x \in \{0,1\}$ if and only if $x^2 = x$

Approach: consider the following pairing relations:

$$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) \text{ and } e\left(\sigma_{\boldsymbol{x}}, \Pi_{i\in[m]}A_i\right)$$

$$e\left([\Sigma_{i\in[m]}\alpha_i x_i], [\Sigma_{i\in[m]}\alpha_i]\right) = [\Sigma_{i\in[m]}\alpha_i^2 x_i] \times [\Sigma_{i\neq j}\alpha_i\alpha_j x_i]$$

non-cross terms *cross terms*

$$e\left([\Sigma_{i\in[m]}\alpha_i x_i], [\Sigma_{i\in[m]}\alpha_i x_i]\right) = [\Sigma_{i\in[m]}\alpha_i^2 x_i^2] \times [\Sigma_{i\neq j}\alpha_i\alpha_j x_i x_j]$$

non-cross terms *cross terms*

Proving Relations on Committed Values

If $x_i^2 = x_i$ for all i, then

$$[\Sigma_{i\in[m]}\alpha_i^2 x_i] = [\Sigma_{i\in[m]}\alpha_i^2 x_i^2]$$

Wire validity

Commitment for each wire is a commitment to a 0/1 vector

$x \in \{0,1\}$ if and only if $x^2 = x$

Approach: consider the following pairing relations:

$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}})$ and $e\left(\sigma_{\boldsymbol{x}}, \Pi_{i\in[m]}A_i\right)$

$$e\left([\Sigma_{i\in[m]}\alpha_i x_i], [\Sigma_{i\in[m]}\alpha_i]\right) \qquad e\left([\Sigma_{i\in[m]}\alpha_i x_i], [\Sigma_{i\in[m]}\alpha_i x_i]\right)$$

When $x_i^2 = x_i$, difference between these terms is

$$[\Sigma_{i\neq j}\alpha_i\alpha_j(x_i - x_i x_j)]$$

Give prover ability to
eliminate cross-terms *only*

Augment CRS with cross-terms

$[\alpha_i\alpha_j]$ $B_{i,j} = g_p^{\alpha_i\alpha_j} \quad \forall i \neq j$

Proving Relations on Committed Values

Prover now computes additional group component in the *base* group

$$\boxed{\left[\Sigma_{i \neq j} \alpha_i \alpha_j \left(x_i - x_i x_j\right)\right]} \xrightarrow{\text{Pair with } g_p} \boxed{\left[\Sigma_{i \neq j} \alpha_i \alpha_j \left(x_i - x_i x_j\right)\right]}$$

$$V = B_{i,j}^{x_i - x_i x_j} \qquad e\left(g_p, V\right)$$

$$e\left(\boxed{\left[\Sigma_{i \in [m]} \alpha_i x_i\right]}, \boxed{\left[\Sigma_{i \in [m]} \alpha_i\right]}\right) \qquad e\left(\boxed{\left[\Sigma_{i \in [m]} \alpha_i x_i\right]}, \boxed{\left[\Sigma_{i \in [m]} \alpha_i x_i\right]}\right)$$

When $x_i^2 = x_i$, difference between these terms is

$$\boxed{\left[\Sigma_{i \neq j} \alpha_i \alpha_j \left(x_i - x_i x_j\right)\right]}$$

Give prover ability to <u>eliminate</u> cross-terms *only* $\longrightarrow$

Augment CRS with cross-terms

$$\boxed{\left[\alpha_i \alpha_j\right]} \quad B_{i,j} = g_p^{\alpha_i \alpha_j} \quad \forall i \neq j$$

Proving Relations on Committed Values

Prover now computes additional group component in the *base* group

$$\left[\Sigma_{i \neq j} \alpha_i \alpha_j \left(x_i - x_i x_j\right)\right] \xrightarrow{\text{Pair with } g_p} \left[\Sigma_{i \neq j} \alpha_i \alpha_j \left(x_i - x_i x_j\right)\right]$$

$$V = B_{i,j}^{x_i - x_i x_j} \qquad\qquad e\left(g_p, V\right)$$

Overall verification relation: $e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e(\sigma_{\boldsymbol{x}}, A) e\left(g_p, V\right)$ $\qquad A = \Pi_{i \in [m]} A_i$

Proving Relations on Committed Values

Prover now computes additional group component in the *base* group

$[\Sigma_{i \neq j} \alpha_i \alpha_j (x_i - x_i x_j)]$ $\xrightarrow{\text{Pair with } g_p}$ $[\Sigma_{i \neq j} \alpha_i \alpha_j (x_i - x_i x_j)]$

$V = B_{i,j}^{x_i - x_i x_j}$ $\qquad$ $e(g_p, V)$

Overall verification relation: $e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e(\sigma_{\boldsymbol{x}}, A) e(g_p, V)$ $\qquad$ $A = \Pi_{i \in [m]} A_i$

Non-cross terms ensure that $x_i^2 = x_i$

Proving Relations on Committed Values

Prover now computes additional group component in the *base* group

$$\left[\Sigma_{i\neq j}\alpha_i\alpha_j\left(x_i - x_i x_j\right)\right] \xrightarrow{\text{Pair with } g_p} \left[\Sigma_{i\neq j}\alpha_i\alpha_j\left(x_i - x_i x_j\right)\right]$$

$$V = B_{i,j}^{x_i - x_i x_j} \qquad\qquad e\left(g_p, V\right)$$

Overall verification relation: $e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e(\sigma_{\boldsymbol{x}}, A)e\left(g_p, V\right)$ $\quad A = \Pi_{i\in[m]}A_i$

Non-cross terms ensure that $x_i^2 = x_i$

Correction factor to correct for cross terms

Proving Relations on Committed Values

Common reference string:

$[\alpha_1]$ $[\cdots]$ $[\alpha_m]$

$A_1 = g_p^{\alpha_1}$ $A_m = g_p^{\alpha_m}$

$[\alpha_1 + \cdots \alpha_m]$ $A = \Pi_{i \in [m]} A_i$

$[\alpha_i \alpha_j]$ $B_{i,j} = g_p^{\alpha_i \alpha_j} \; \forall i \neq j$

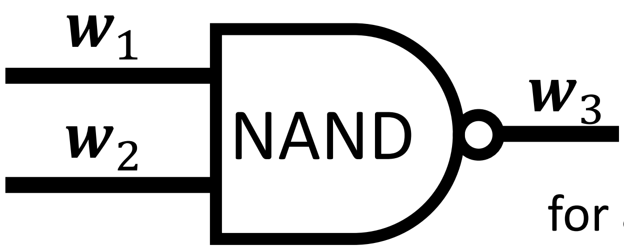
Commitment to $(x_1, \ldots, x_m)$:

$[\Sigma_{i \in [m]} \alpha_i x_i]$

$$\sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m} = g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}$$

Gate validity

For each gate, commitment to output wires is consistent with gate operation and commitment to input wires



for all $i \in [m]$: $w_{3,i} = 1 - w_{1,i} w_{2,i}$

Can leverage same approach as before:

$$e(\sigma_{w_3}, A) = e(g_p, g_p)^{\Sigma_{i \in [m]} \alpha_i^2 w_{3,i} + \Sigma_{i \neq j} \alpha_i \alpha_j w_{3,i}}$$

$$e(A, A) = e(g_p, g_p)^{\Sigma_{i \in [m]} \alpha_i^2 + \Sigma_{i \neq j} \alpha_i \alpha_j}$$

$$e(\sigma_{w_1}, \sigma_{w_2}) = e(g_p, g_p)^{\Sigma_{i \in [m]} \alpha_i^2 w_{1,i} w_{2,i} + \Sigma_{i \neq j} \alpha_i \alpha_j w_{1,i} w_{2,j}}$$

Proving Relations on Committed Values

Common reference string:

$[\alpha_1]$ $[\cdots]$ $[\alpha_m]$

$A_1 = g_p^{\alpha_1}$ $A_m = g_p^{\alpha_m}$

$[\alpha_1 + \cdots \alpha_m]$ $A = \Pi_{i \in [m]} A_i$

$[\alpha_i \alpha_j]$ $B_{i,j} = g_p^{\alpha_i \alpha_j} \; \forall i \neq j$

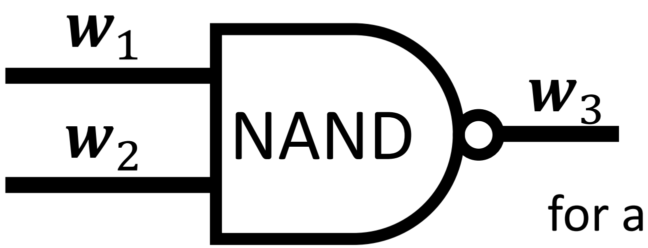
If $w_{3,i} + w_{1,i} w_{2,i} = 1$ for all i, then

$$\frac{e(\sigma_{w_3}, A) e(\sigma_{w_1}, \sigma_{w_2})}{e(A, A)}$$

only consists of cross terms!

Gate validity

For each gate, commitment to output wires is consistent with gate operation and commitment to input wires



for all $i \in [m]$: $w_{3,i} = 1 - w_{1,i} w_{2,i}$

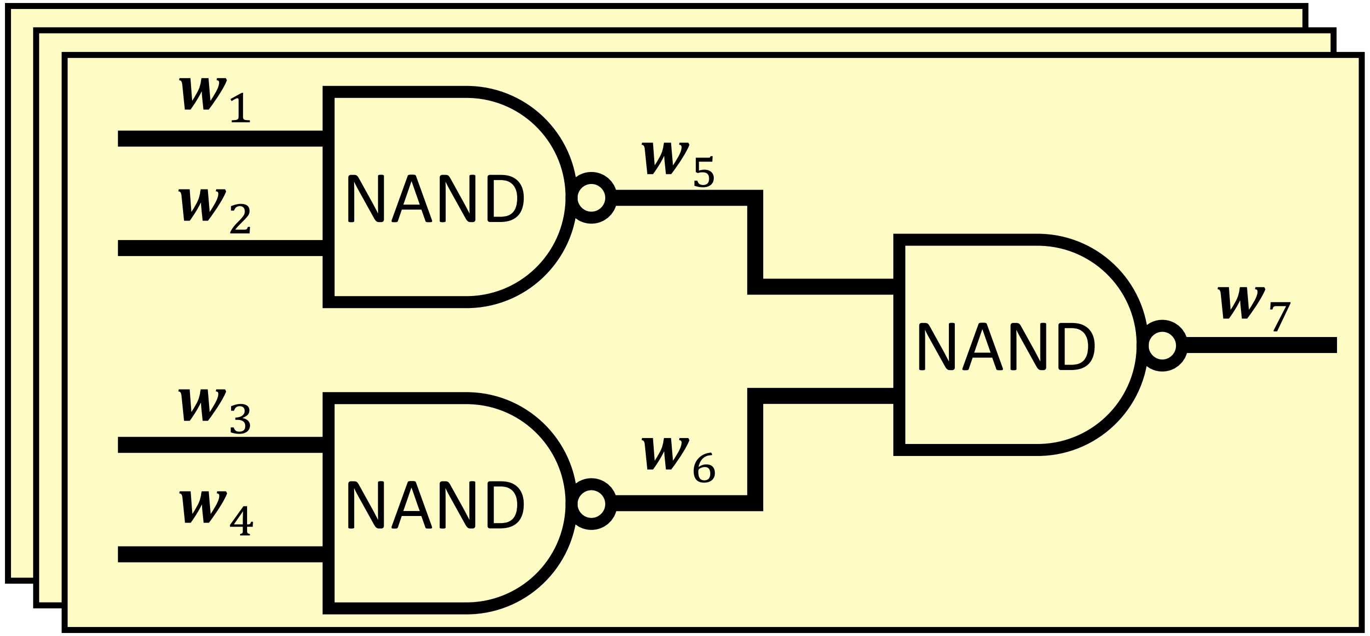
Can leverage same approach as before:

$$e(\sigma_{\boldsymbol{w}_3}, A) = e(g_p, g_p)^{\sum_{i \in [m]} \alpha_i^2 w_{3,i} + \sum_{i \neq j} \alpha_i \alpha_j w_{3,i}}$$

$$e(A, A) = e(g_p, g_p)^{\sum_{i \in [m]} \alpha_i^2 + \sum_{i \neq j} \alpha_i \alpha_j}$$

$$e(\sigma_{\boldsymbol{w}_1}, \sigma_{\boldsymbol{w}_2}) = e(g_p, g_p)^{\sum_{i \in [m]} \alpha_i^2 w_{1,i} w_{2,i} + \sum_{i \neq j} \alpha_i \alpha_j w_{1,i} w_{2,j}}$$

Proof Size



Let $\boldsymbol{w}_i = (w_{i,1}, \ldots, w_{i,m})$ be vector of wire labels associated with wire i

1 Prover commits to each vector of wire assignments

$$\boldsymbol{w}_i = \boxed{w_{i,1}}\ \boxed{w_{i,2}}\ \boxed{\cdots}\ \boxed{w_{i,m}} \rightarrow \boxed{\sigma_i}$$

2 Prover constructs the following proofs:

Input validity

Wire validity One group element

Gate validity One group element

Output validity

Commitment size: $|\sigma_i| = \text{poly}(\lambda)$

Single group element

Overall proof size (t wires, s gates):

$$(2t + s) \cdot \text{poly}(\lambda) = |C| \cdot \text{poly}(\lambda)$$

Is This Sound?

Common reference string:

$[\alpha_1]$ $[\cdots]$ $[\alpha_m]$

$A_1 = g_p^{\alpha_1}$ $A_m = g_p^{\alpha_m}$

$[\alpha_1 + \cdots \alpha_m]$ $A = \Pi_{i\in[m]} A_i$

$[\alpha_i \alpha_j]$ $B_{i,j} = g_p^{\alpha_i \alpha_j}\ \forall i \neq j$

Commitment to $(x_1, \dots, x_m)$:

$[\Sigma_{i\in[m]} \alpha_i x_i]$

$$\sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m} = g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}$$

Soundness requires some care:

Groth-Ostrovsky-Sahai NIZK based on similar commit-and-prove strategy

Soundness in GOS is possible by ***extracting*** a witness from the commitment

For a false statement, no witness exists

Our setting: commitments are *succinct* – cannot extract a full witness

Solution: "local extractability" [KPY19] or "somewhere extractability" [CJJ21]

Somewhere Soundness

CRS will have two modes:

Normal mode: used in the real scheme

Extracting on index $\boldsymbol{i}$: supports witness extraction for instance i (given a trapdoor)

If proof π verifies, then we can extract a witness w_i such that $C(x_i, w_i) = 1$

CRS in the two modes are computationally indistinguishable

Similar to "dual-mode" proof systems and somewhere statistically binding hash functions

Implies non-adaptive soundness

Local Extraction

	A_1		A_{i^*-1}	A_{i^*}	A_{i^*+1}		A_m
Normal mode:	$g_p^{\alpha_1}$	$\cdots$	$g_p^{\alpha_{i^*-1}}$	$g_p^{\alpha_{i^*}}$	$g_p^{\alpha_{i^*+1}}$	$\cdots$	$g_p^{\alpha_m}$

Move slot i^* to full group

Extracting mode: (extract on i^*)	$g_p^{\alpha_1}$	$\cdots$	$g_p^{\alpha_{i^*-1}}$	$g_p^{\alpha_{i^*}} g_q^r$	$g_p^{\alpha_{i^*+1}}$	$\cdots$	$g_p^{\alpha_m}$
	A_1		A_{i^*-1}	A_{i^*}	A_{i^*+1}		A_m

Subgroup decision assumption [BGN05]:

Random element in subgroup ($\mathbb{G}_p$)

$\approx$

Random element in full group ($\mathbb{G}$)

Local Extraction

CRS in extraction mode (for index i^*):

A_1		A_{i^*-1}	A_{i^*}	A_{i^*+1}		A_m
$g_p^{\alpha_1}$	$\cdots$	$g_p^{\alpha_{i^*-1}}$	$g_p^{\alpha_{i^*}} g_q^r$	$g_p^{\alpha_{i^*+1}}$	$\cdots$	$g_p^{\alpha_m}$

Trapdoor: g_q (generator of $\mathbb{G}_q$)

Can extract by projecting into $\mathbb{G}_q$

- Extracted bit for a commitment $\boldsymbol{\sigma}$ is 1 if $\boldsymbol{\sigma}$ has a (non-zero) component in $\mathbb{G}_q$

Correctness of Extraction

Consider wire validity check:

$$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e(\sigma_{\boldsymbol{x}}, A)e(g_p, V)$$

Correctness of Extraction

Consider wire validity check:

$$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e(\sigma_{\boldsymbol{x}}, A)e(g_p, V)$$

Adversary chooses commitment $\sigma_{\boldsymbol{x}}$ and proof V

Correctness of Extraction

Consider wire validity check:

$$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e(\sigma_{\boldsymbol{x}}, A)e(g_p, V)$$

Adversary chooses commitment $\sigma_{\boldsymbol{x}}$ and proof V

Generator g_p and aggregated component A part of the CRS (honestly-generated)

If this relation holds, it must hold in **both** the order-p subgroup **and** the order-q subgroup of $\mathbb{G}_T$

Key property: $e(g_p, V)$ is **always** in the order-p subgroup; adversary **cannot** influence the verification relation in the order-q subgroup

Write $\sigma_{\boldsymbol{x}} = g_p^s g_q^t$

Write $A = g_p^{\sum_{i\in[m]} \alpha_i} g_q^r$

In the order-q subgroup, exponents must satisfy:

$$t^2 = tr \bmod q$$

Correctness of Extraction

Consider wire validity check:

$$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e(\sigma_{\boldsymbol{x}}, A)e(g_p, V)$$

Adversary chooses commitment $\sigma_{\boldsymbol{x}}$ and proof V

Generator g_p and aggregated component A part of the CRS (honestly-generated)

If this relation holds, it must hold in **both**
the order-p subgroup **and** the order-q subgroup of $\mathbb{G}$

If wire validity checks pass, then $t = b_i r$ where $b_i \in \{0,1\}$

Observe: $b_i \in \{0,1\}$ is also the extracted bit

Key property: $e(g_p, V)$ is **alwa**
verification relation in the ord

Write $\sigma_{\boldsymbol{x}} = g_p^s g_q^t$

Write $A = g_p^{\sum_{i\in[m]} \alpha_i} g_q^r$

In the order-q subgroup, exponents must satisfy:

$$t^2 = tr \bmod q$$

Correctness of Extraction

Consider gate validity check:

$$e(\sigma_{\boldsymbol{w}_3}, A)e(\sigma_{\boldsymbol{w}_1}, \sigma_{\boldsymbol{w}_2}) = e(A, A)e(g_p, W)$$

Correctness of Extraction

Consider gate validity check:

$$e(\sigma_{\boldsymbol{w}_3}, A)e(\sigma_{\boldsymbol{w}_1}, \sigma_{\boldsymbol{w}_2}) = e(A, A)e(g_p, W)$$

Adversary chooses commitment $\sigma_{\boldsymbol{w}_1}, \sigma_{\boldsymbol{w}_2}, \sigma_{\boldsymbol{w}_3}$ and proof W

Generator g_p and aggregated key A part of the CRS (honestly-generated)

Write

$$\sigma_{\boldsymbol{w}_1} = g_p^{s_1} g_q^{t_1}$$
$$\sigma_{\boldsymbol{w}_2} = g_p^{s_2} g_q^{t_2}$$
$$\sigma_{\boldsymbol{w}_3} = g_p^{s_3} g_q^{t_3}$$

Write $A = g_p^{\sum_{i \in [m]} \alpha_i} g_q^r$

In the order-q subgroup, exponents must satisfy:

$$t_3 r + t_1 t_2 = r^2 \bmod q$$

By wire validity checks: $t_i = b_i r$ where $b_i \in \{0,1\}$

$$b_3 r^2 + b_1 b_2 r^2 = r^2 \bmod q$$

$$b_3 = 1 - b_1 b_2 = \mathrm{NAND}(b_1, b_2)$$

Correctness of Extraction

Consider gate validity check:

$$e(\sigma_{\boldsymbol{w}_3}, A)e(\sigma_{\boldsymbol{w}_1}, \sigma_{\boldsymbol{w}_2}) = e(A, A)e(g_p, W)$$

Adversary chooses commitment $\sigma_{\boldsymbol{w}_1}, \sigma_{\boldsymbol{w}_2}, \sigma_{\boldsymbol{w}_3}$ and proof W

Generator g_p and aggregated key A part of the CRS (honestly-generated)

Write

$$\sigma_{w_1} = g_p^{s_1} g_q^{t_1}$$
$$\sigma_{w_2} = g_p^{s_2} g_q^{t_2}$$
$$\sigma_{w_3} = g_p^{s_3} g_q^{t_3}$$

Write $A = g_p^{\sum_{i \in [m]} \alpha_i} g_q^r$

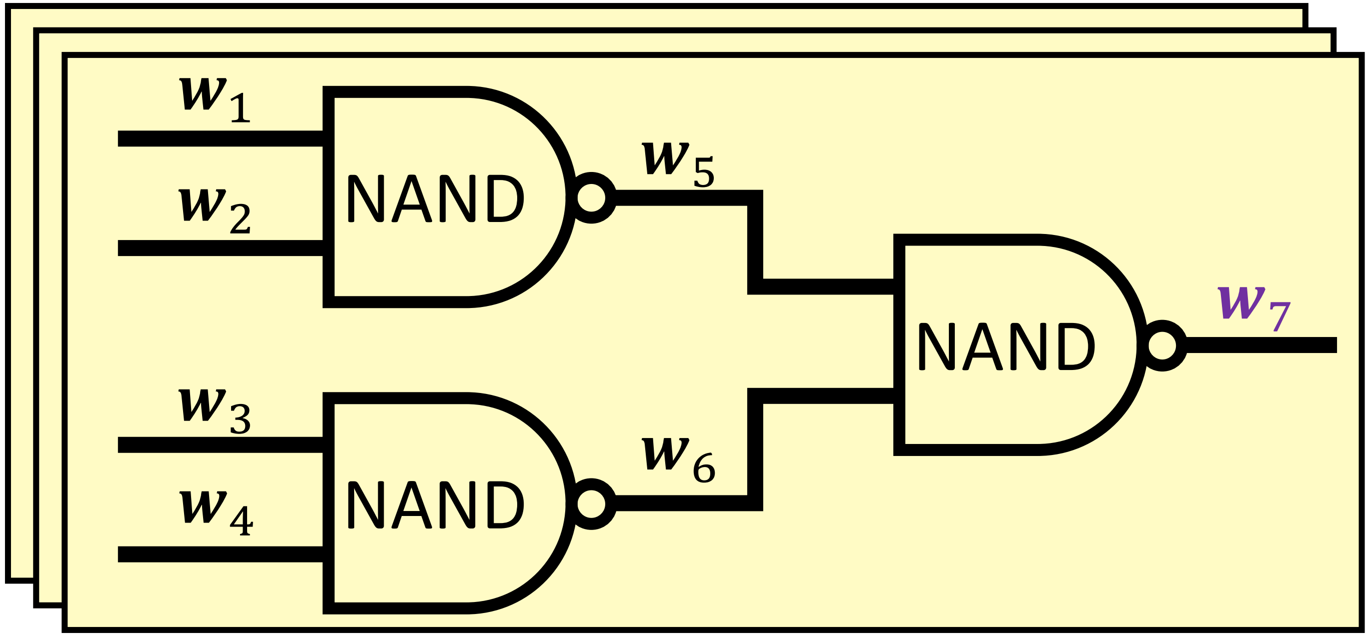
In the order-q subgroup, exponents must satisfy:

$$t_3 r + t_1 t_2 = r^2 \bmod q$$

Conclusion: extracted bits are consistent with gate operation

$$b_3 = 1 - b_1 b_2 = \mathrm{NAND}(b_1, b_2)$$

A Commit-and-Prove Strategy for Batch Arguments



Let $\boldsymbol{w}_i = (w_{i,1}, \ldots, w_{i,m})$ be vector of wire labels associated with wire i across the m instances

1 Prover commits to each vector of wire assignments

2 Prover constructs the following proofs:

Input validity

Wire validity

Gate validity

Output validity

Key idea: Validity checks are quadratic and can be checked in the exponent

From Composite-Order to Prime-Order

Batch arguments for NP from standard assumptions over bilinear maps

Subgroup decision assumption in composite-order bilinear groups

$$\mathbb{G} \cong \mathbb{G}_p \times \mathbb{G}_q$$

composite-order group

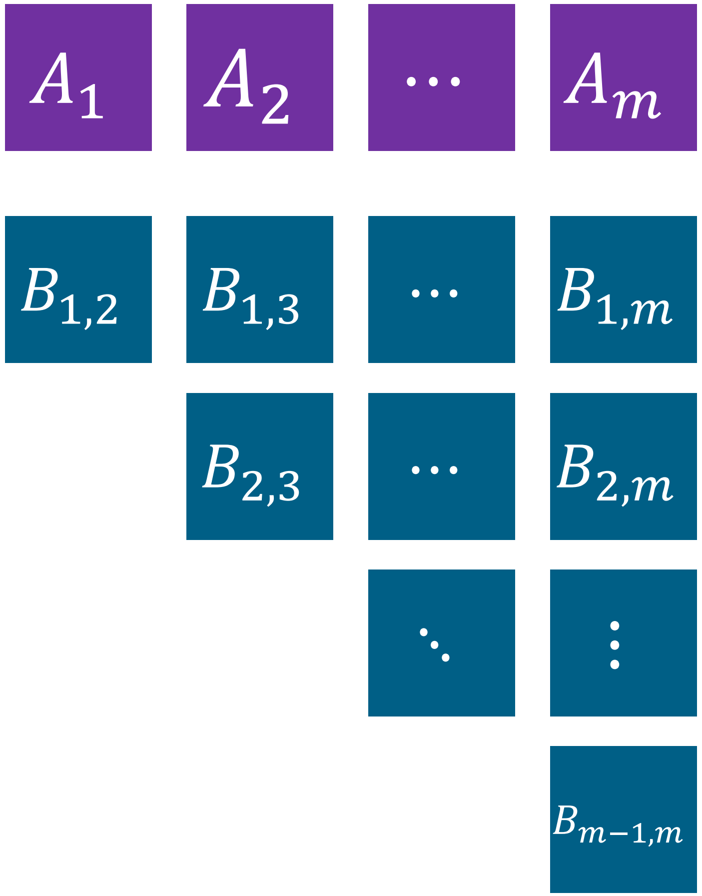
Simulate subgroups with subspaces

Yields a batch argument from

k-Linear assumption (for any $k \geq 1$) in prime-order asymmetric bilinear groups

Reducing CRS Size

Common reference string:



Size of CRS is $m^2 \cdot \mathrm{poly}(\lambda)$

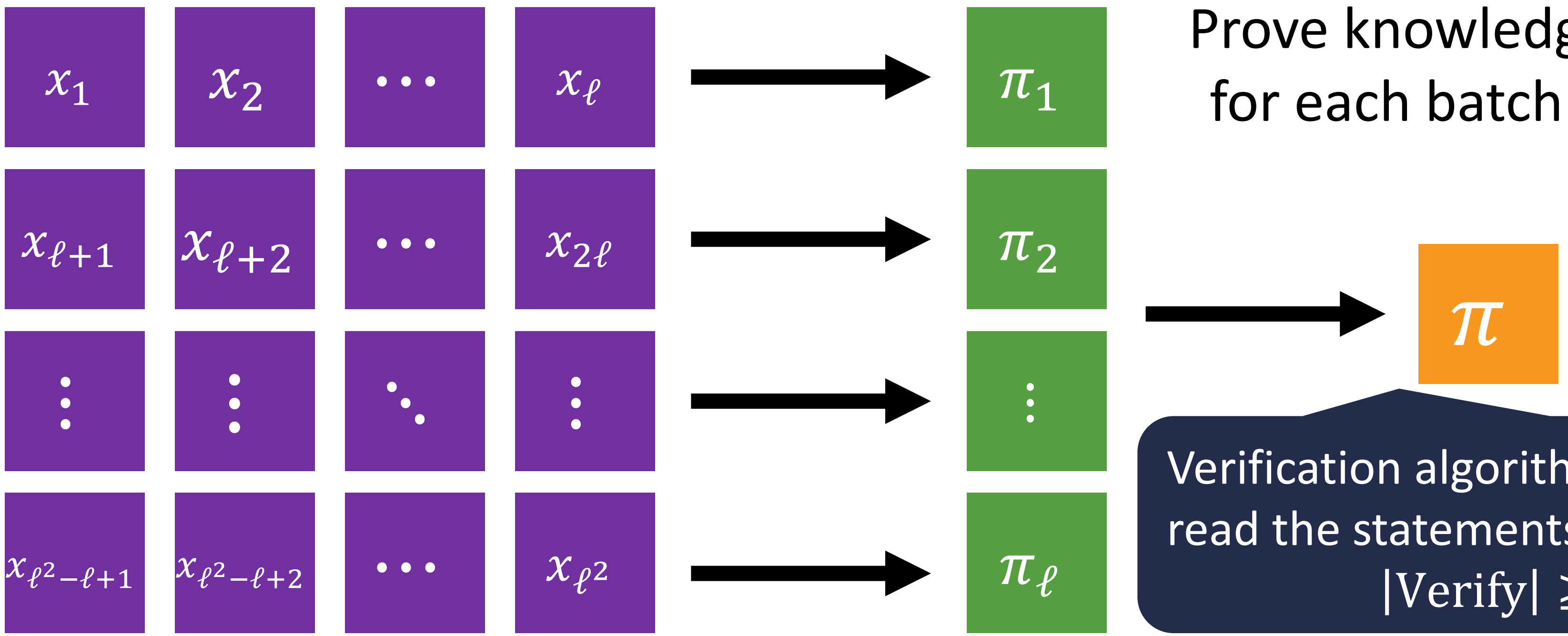
Can rely on recursive composition to reduce CRS size:

$$m^2 \cdot \mathrm{poly}(\lambda) \to m^{\varepsilon} \cdot \mathrm{poly}(\lambda)$$

for any constant $\varepsilon > 0$

Similar approach as [KPY19]

The Base Case



Prove knowledge of a proof π_i for each batch of statements

Verification algorithm for a batch needs to read the statements (of length ℓ), so

$$|\text{Verify}| \geq \sqrt{m} \cdot \text{poly}(\lambda)$$

$$\ell = \sqrt{m}$$

Use batch argument on $\ell = \sqrt{m}$ instances to prove each batch

Both batch arguments are on $\ell = \sqrt{m}$ statements

Batch Arguments with Split Verification

$$\mathrm{Verify}(\mathrm{crs}, C, (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m), \pi)$$

$$\mathrm{GenVK}\big(\mathrm{crs}, (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m)\big) \to \mathrm{vk}$$

Runs in time $\mathrm{poly}(\lambda, m, n)$

$|\mathrm{vk}| = \mathrm{poly}(\lambda, \log m, n)$

Preprocesses statements into a <u>short</u> verification key

$$\mathrm{OnlineVerify}(\mathrm{vk}, C, \pi)$$

Runs in time $\mathrm{poly}(\lambda, \log m, |C|)$

Fast online verification

(Similar property from [CJJ21])

Recursive Bootstrapping

x_1	x_2	$\cdots$	x_ℓ	$\longrightarrow$	π_1
$x_{\ell+1}$	$x_{\ell+2}$	$\cdots$	$x_{2\ell}$	$\longrightarrow$	π_2
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\longrightarrow$	$\vdots$
$x_{\ell^2-\ell+1}$	$x_{\ell^2-\ell+2}$	$\cdots$	x_{ℓ^2}	$\longrightarrow$	π_ℓ

$$\ell = \sqrt{m}$$

Use batch argument on $\ell = \sqrt{m}$ instances to prove each batch

Prove knowledge of a proof π_i for each batch of statements

$\pi_1, \pi_2, \ldots, \pi_\ell \longrightarrow \pi$

Overall proof size: $\mathrm{poly}(\lambda, \log m, |C|)$

CRS size: $m \cdot \mathrm{poly}(\lambda)$

Batch argument used to check the relation

$$\mathcal{R}\big((C, \mathrm{vk}_1, \ldots, \mathrm{vk}_\ell), (\pi_1, \ldots, \pi_\ell)\big) = 1$$

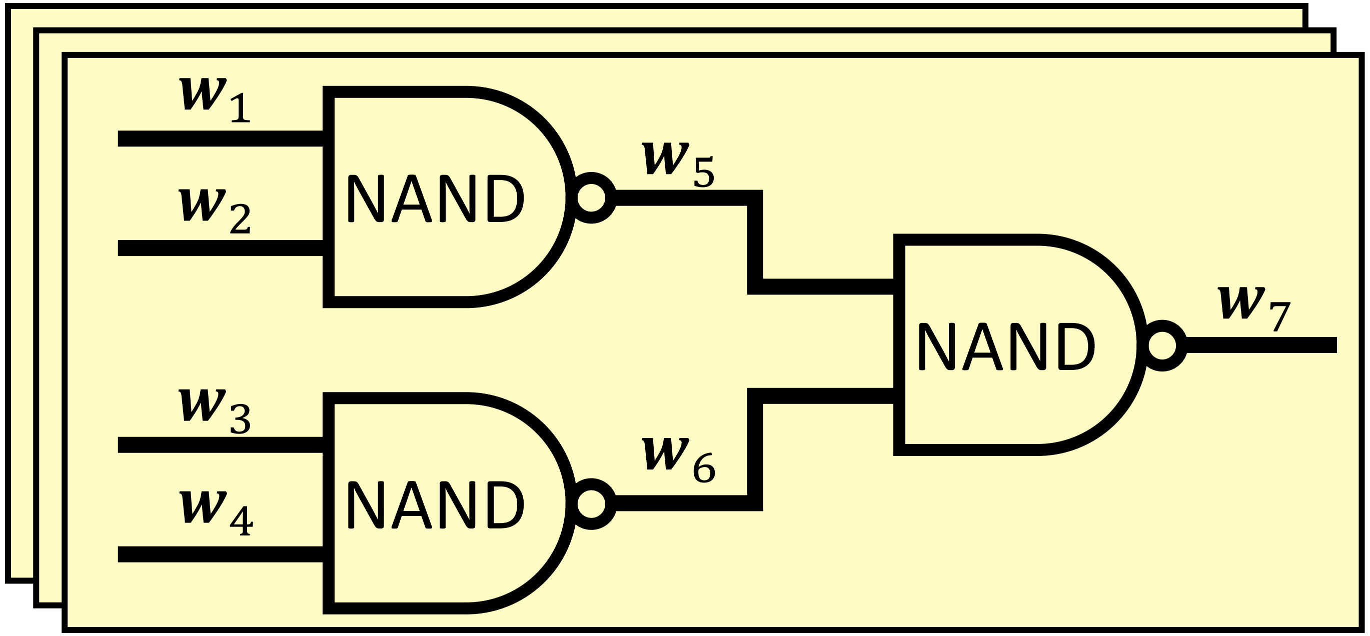
if $\mathrm{OnlineVerify}(\mathrm{vk}_i, C, \pi_i) = 1$

$$|\mathrm{OnlineVerify}| = \mathrm{poly}(\lambda, \log m, |C|)$$

Both batch arguments are on $\ell = \sqrt{m}$ statements

After $k \approx \log 1/\varepsilon$ steps $\Rightarrow m^\varepsilon \cdot \mathrm{poly}(\lambda)$ size CRS

Batch Arguments with Split Verification



In online phase, verifier uses commitments $(\sigma_1, \dots, \sigma_n)$ for the bits of input wires

(no more input validity checks)

Verifier checks the following

Input validity } $nm \cdot \text{poly}(\lambda)$

Wire validity

Gate validity

Output validity

} $|C| \cdot \text{poly}(\lambda)$

constant number of group operations per wire/gate

Only depends on the statement!

Given $(\boldsymbol{x}_1, \dots, \boldsymbol{x}_m) \in (\{0,1\}^n)^m$, verifier computes commitments to bits of the statement

$$\forall j \in [n] : \ \sigma_j \leftarrow \prod_{i \in [m]} A_i^{x_{i,j}}$$

$$\text{GenVK}\big(\text{crs}, (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m)\big) \to (\sigma_1, \dots, \sigma_n)$$

Batch Arguments with Short CRS

Corollary: Batch arguments for NP from standard assumptions over bilinear maps

k-Linear assumption (for any $k \geq 1$) in prime-order bilinear groups

Subgroup decision assumption in composite-order bilinear groups

For a proof on m instances of length n:

- **CRS size:** $|\mathrm{crs}| = m^{\varepsilon} \cdot \mathrm{poly}(\lambda)$ for any constant $\varepsilon > 0$
- **Proof size:** $|\pi| = \mathrm{poly}(\lambda, |C|)$
- **Verification time:** $|\mathrm{Verify}| = \mathrm{poly}(\lambda, n, m) + \mathrm{poly}(\lambda, |C|)$

Application to RAM Delegation ("SNARGs for P")

Choudhuri et al. [CJJ21] showed:

succinct argument for polynomial-time computations

Batch argument with split verification + Somewhere extractable commitment → Delegation scheme for RAM programs

succinct vector commitment that allows extracting on single index

Application to RAM Delegation ("SNARGs for P")

Choudhuri et al. [CJJ21] showed:

Batch argument with split verification + Somewhere extractable commitment → Delegation scheme for RAM programs

succinct argument for polynomial-time computations

This work (from k-Lin)

succinct vector commitment that allows extracting on single index

Application to RAM Delegation (“SNARGs for P”)

Choudhuri et al. [CJJ21] showed:

Batch argument with split verification	+	Somewhere extractable commitment	→	Delegation scheme for RAM programs
This work (from k-Lin)		*This work + [OPWW15] (from SXDH)*		

Application to RAM Delegation ("SNARGs for P")

Choudhuri et al. [CJJ21] showed:

Batch argument with split verification + Somewhere extractable commitment $\Rightarrow$ Delegation scheme for RAM programs

This work (from k-Lin)

This work + [OPWW15] (from SXDH)

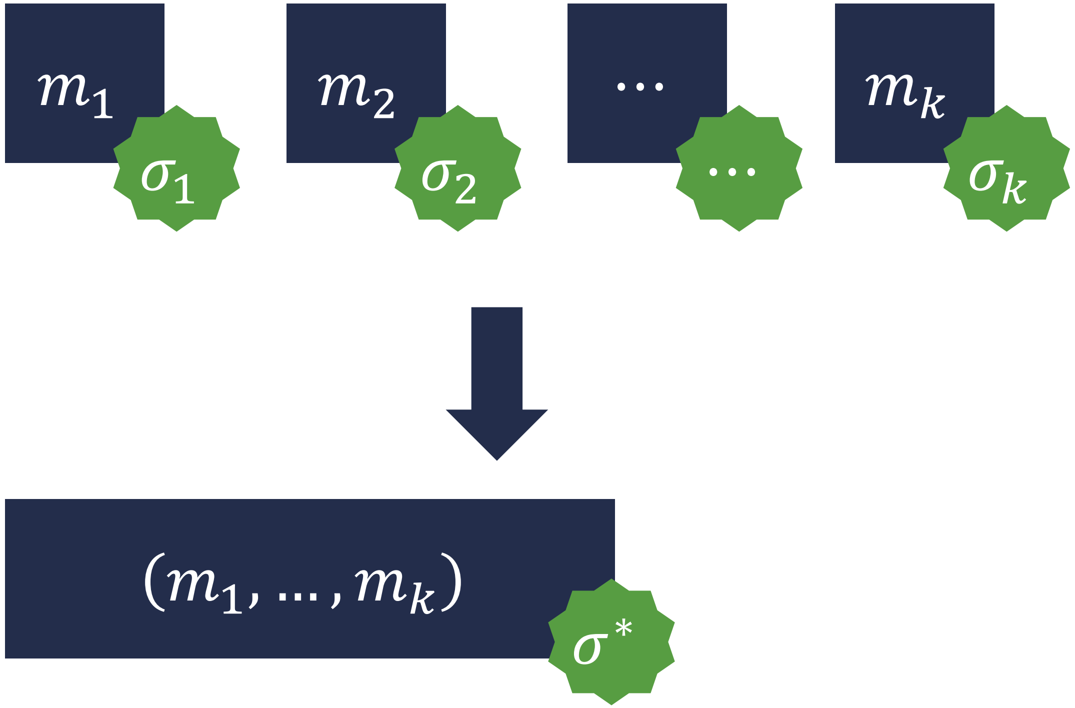
Corollary. RAM delegation from SXDH on prime-order pairing groups

To verify a time-T RAM computation:

- **CRS size:** $|\text{crs}| = T^{\varepsilon} \cdot \text{poly}(\lambda)$ for any constant $\varepsilon > 0$
- **Proof size:** $|\pi| = \text{poly}(\lambda, \log T)$
- **Verification time:** $|\text{Verify}| = \text{poly}(\lambda, \log T)$

Previous pairing constructions: non-standard assumptions [KPY19] or quadratic CRS [GZ21]

Application to Aggregate Signatures



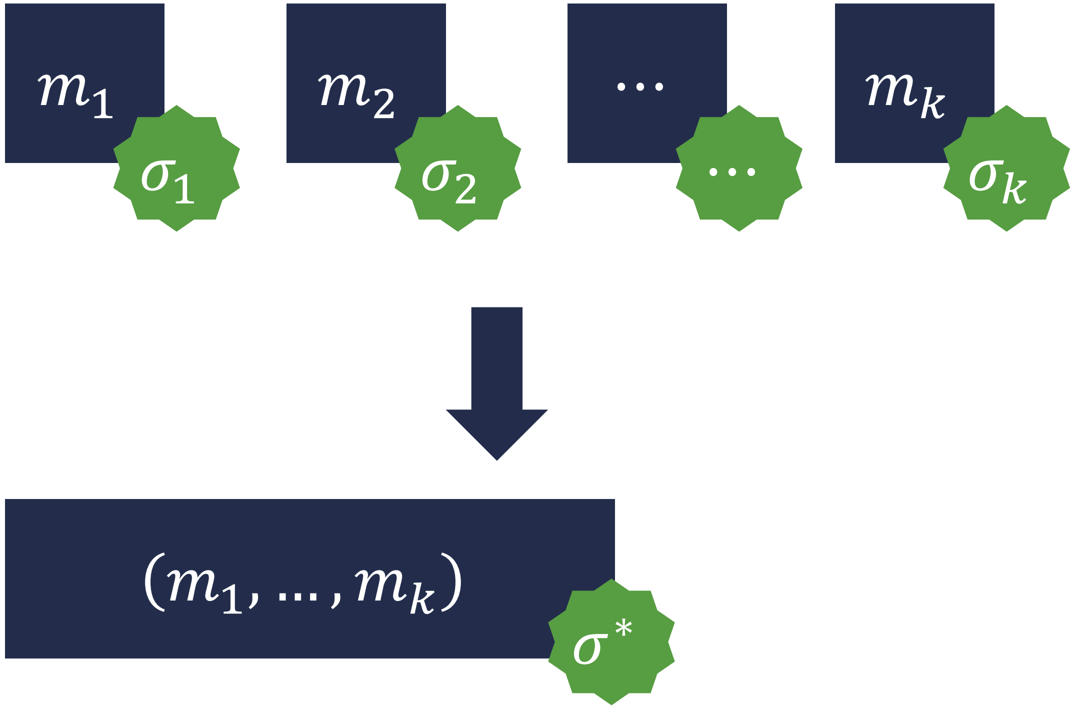
Given k message-signature pairs (m_i, σ_i)

Short signature σ^* on $(m_1, \dots, m_k)$:
$|\sigma^*| = \mathrm{poly}(\lambda, \log k)$

Folklore construction from succinct arguments for NP (SNARKs for NP):

prove knowledge of $\sigma_1, \dots, \sigma_k$ such that $\mathrm{Verify}(\mathrm{vk}, m_i, \sigma_i) = 1$

Application to Aggregate Signatures



Given k message-signature pairs (m_i, σ_i)

Short signature σ^* on $(m_1, \dots, m_k)$:

$$|\sigma^*| = \mathrm{poly}(\lambda, \log k)$$

Can replace SNARKs for NP with a batch argument for NP:

prove knowledge of $\sigma_1, \dots, \sigma_k$ such that $\mathrm{Verify}(\mathrm{vk}, m_i, \sigma_i) = 1$

Application to Aggregate Signatures

Can replace SNARKs for NP with a batch argument for NP:

prove knowledge of $\sigma_1, \ldots, \sigma_k$ such that $\mathrm{Verify}(\mathrm{vk}, m_i, \sigma_i) = 1$

This work: Batch argument for <u>bounded</u> number of instances

Corollary. Aggregate signature supporting <u>bounded</u> aggregation from bilinear maps

First aggregate signature with bounded aggregation from standard pairing-based assumptions (i.e., k-Lin) in the plain model

Previous pairing constructions: unbounded aggregation from standard pairing-based assumptions in the random oracle model [BGLS03]

Summary

Batch arguments for NP from standard assumptions over bilinear maps

Key feature: Construction is “low-tech”

Direct “commit-and-prove” approach like classic pairing-based proof systems

Corollary: RAM delegation (i.e., “SNARG for P”) with sublinear CRS

Corollary: Aggregate signature with bounded aggregation

`https://eprint.iacr.org/2022/336`

Thank you!