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

$(x_1, \ldots, x_m)$

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

verifier

Naïve solution: send witnesses $w_1, \ldots, w_m$ 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

$(x_1, \ldots, x_m)$

π

verifier

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

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

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

$\pi \longrightarrow$

verifier

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

In general setting, verifier needs to read statements

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

Batch Arguments for NP (BARGs)

This work: New constructions of non-interactive 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

BARGs from correlation intractable hash functions

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

Learning with errors (LWE) [CJJ21b]

BARGs 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

BARGs 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 construction à la classic NIZK construction of Groth-Ostrovsky-Sahai

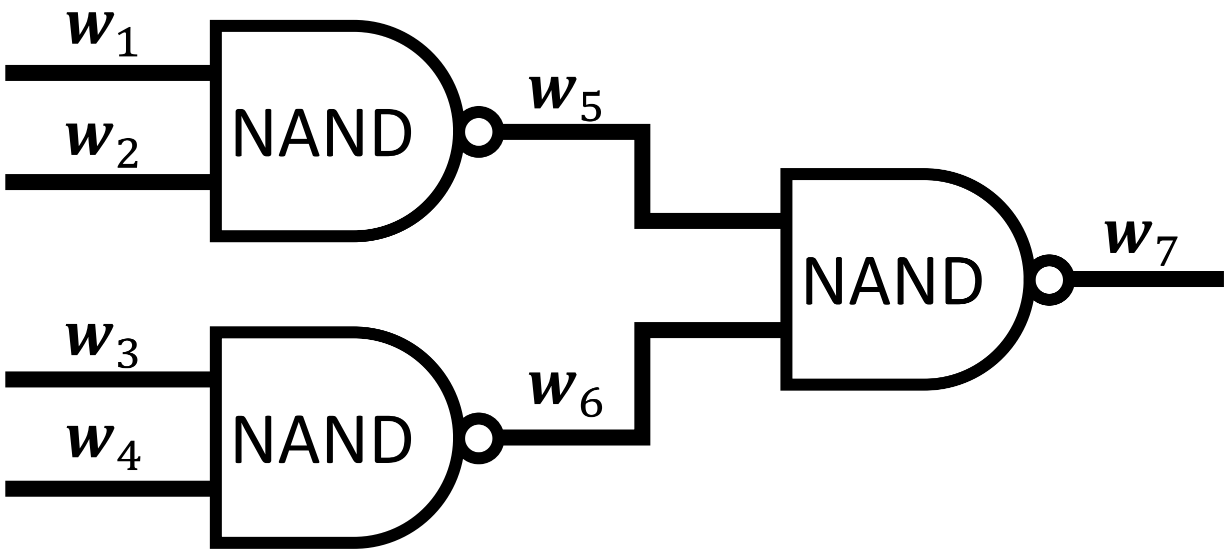
Corollary: RAM delegation (i.e., "SNARG for P") with sublinear CRS from standard bilinear map assumptions

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 BARGs



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

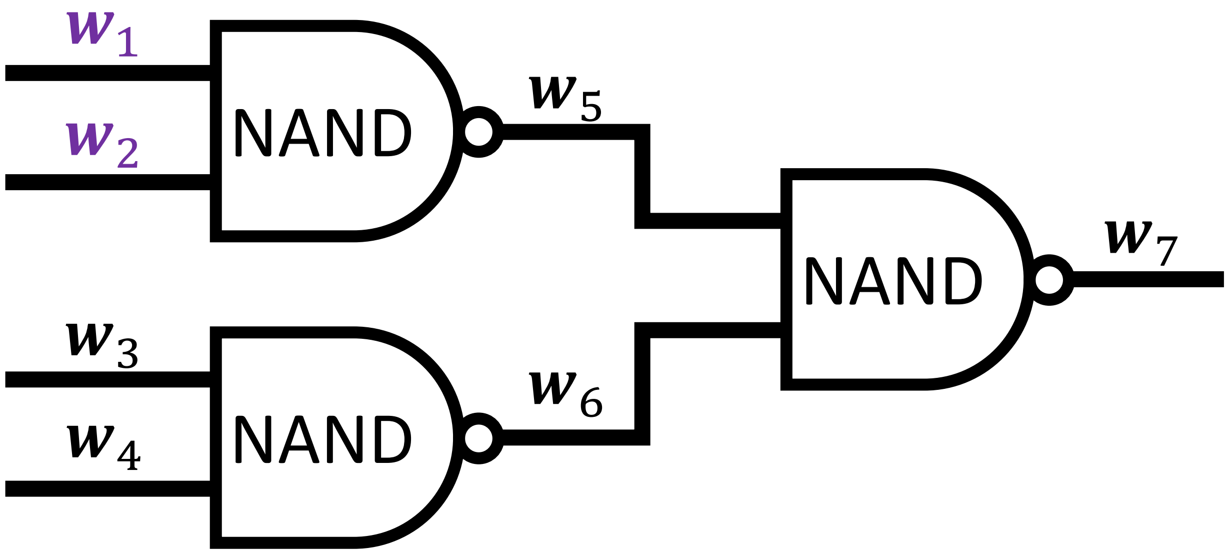
1 Prover commits to each vector of wire assignments

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

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

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

A Commit-and-Prove Strategy for BARGs



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

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

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

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

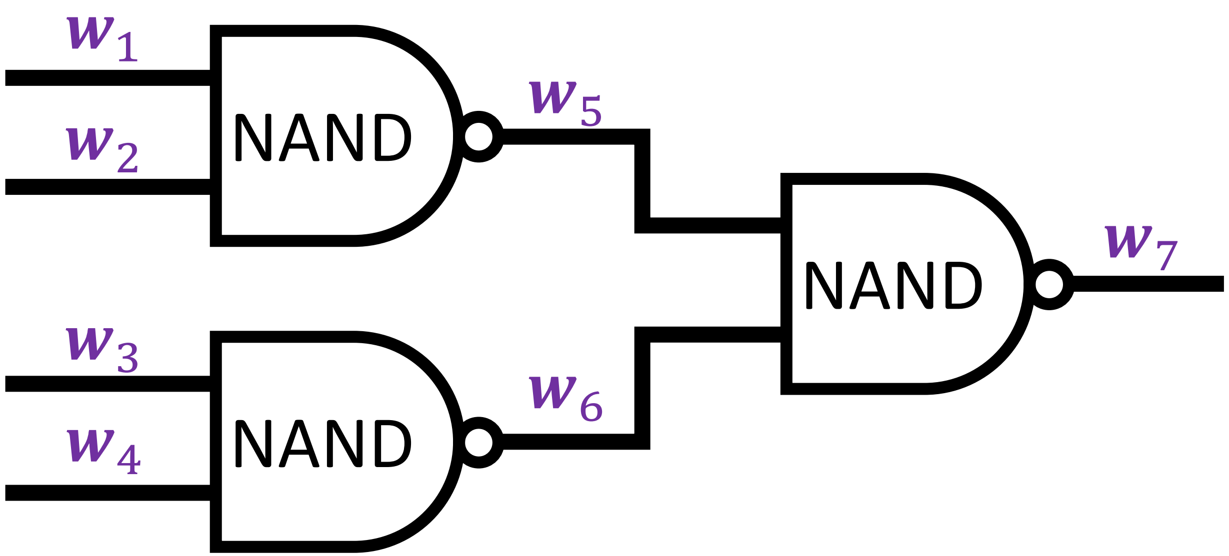
2 Prover constructs the following proofs:

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 BARGs



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

1 Prover commits to each vector of wire assignments

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

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

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

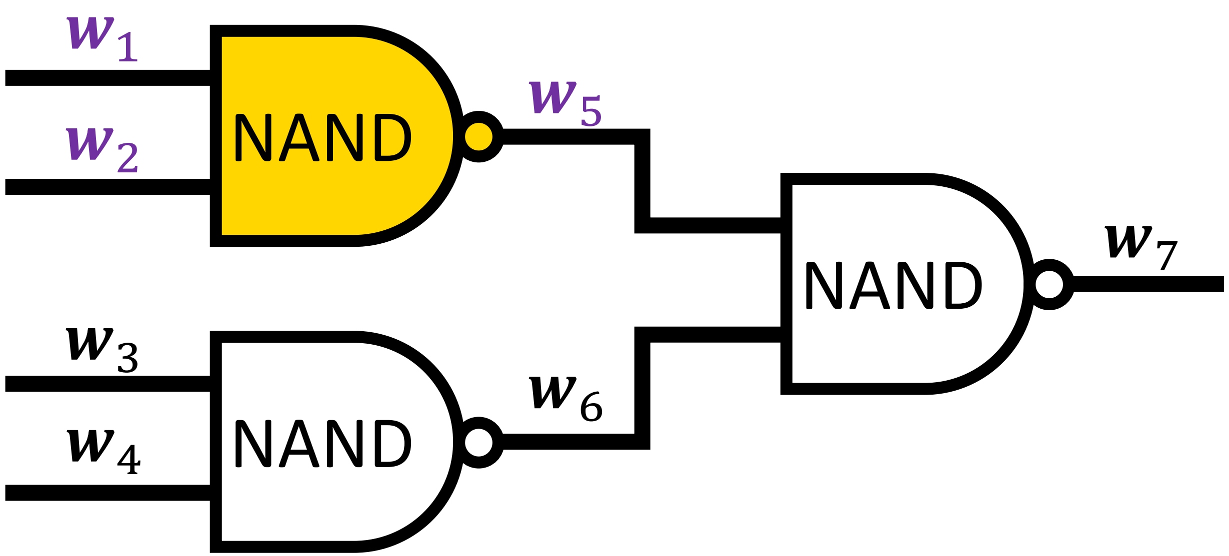
2 Prover constructs the following proofs:

Input validity

Wire validity

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

A Commit-and-Prove Strategy for BARGs



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

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

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

2 Prover constructs the following proofs:

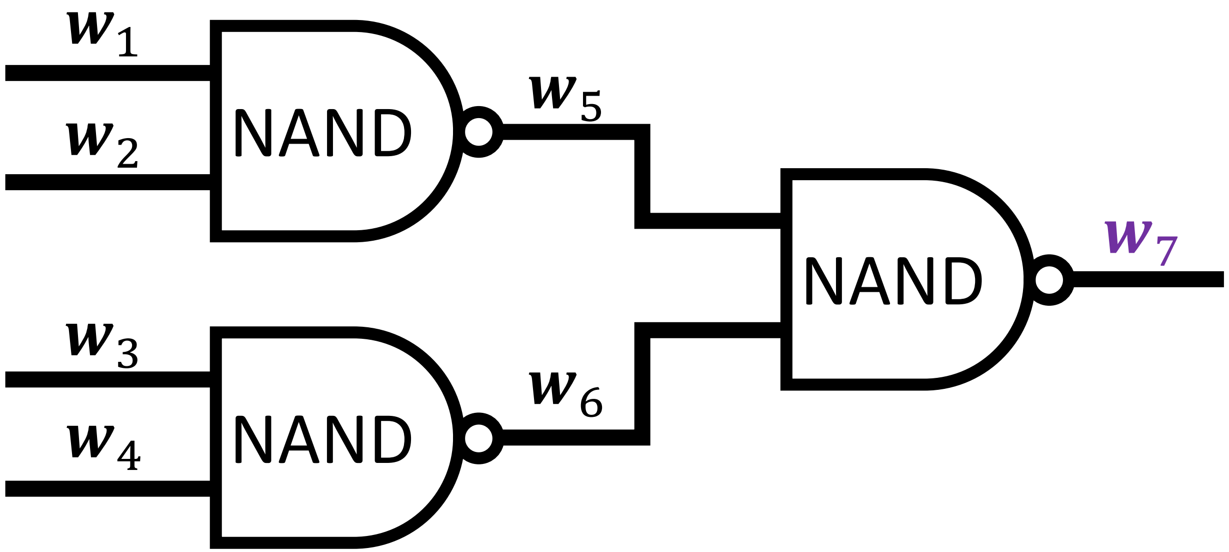
Input validity

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 BARGs



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

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

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

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

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)

Proving Relations on Committed Values

common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$

commitment to $(x_1, \dots, x_m)$

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

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 $\mathbb{G}$:

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

$$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e\left(g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}, g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}\right)$$
$$= e(g_p, g_p)^{(\alpha_1 x_1 + \cdots \alpha_m x_m)^2}$$

Consider the exponent:

$$(\alpha_1 x_1 + \cdots + \alpha_m x_m)^2 = \sum_{i \in [m]} \alpha_i^2 x_i^2 + \sum_{i \neq j} \alpha_i \alpha_j x_i x_j$$

Proving Relations on Committed Values

common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$

commitment to $(x_1, \dots, x_m)$

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

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 <u>efficiently-computable</u> bilinear map on $\mathbb{G}$:

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

$$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e\left(g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}, g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}\right)$$
$$= e(g_p, g_p)^{(\alpha_1 x_1 + \cdots \alpha_m x_m)^2}$$

Consider the exponent:

$$(\alpha_1 x_1 + \cdots + \alpha_m x_m)^2 = \sum_{i \in [m]} \alpha_i^2 x_i^2 + \sum_{i \neq j} \alpha_i \alpha_j x_i x_j$$

cross-terms

Proving Relations on Committed Values

common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$

commitment to $(x_1, \dots, x_m)$

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

If $x_i^2 = x_i$ for all i, then these expressions are equal up to cross-terms

If $x_1, \dots, x_m \in \{0,1\}$, then $x_i^2 = x_i$ and

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

Let $A = A_1 A_2 \cdots A_m = g_p^{\sum_{i \in [m]} \alpha_i}$

Next:

$$(\alpha_1 x_1 + \cdots + \alpha_m x_m)(\alpha_1 + \cdots + \alpha_m) = \sum_{i \in [m]} \alpha_i^2 x_i + \sum_{i \neq j} \alpha_i \alpha_j x_i$$

Consider the exponent:

$$(\alpha_1 x_1 + \cdots + \alpha_m x_m)^2 = \sum_{i \in [m]} \alpha_i^2 x_i^2 + \underbrace{\sum_{i \neq j} \alpha_i \alpha_j x_i x_j}_{\text{cross-terms}}$$

Proving Relations on Committed Values

common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$

$$\forall i \neq j: B_{ij} = g_p^{\alpha_i \alpha_j}$$

Approach: augment CRS with cross-terms

$$A = g_p^{\alpha_1 + \cdots + \alpha_m}$$

commitment to $(x_1, \ldots, x_m)$

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

If $x_i^2 = x_i$ for all i, then these expressions are equal up to cross-terms

If $x_1, \ldots, x_m \in \{0,1\}$, then $x_i^2 = x_i$ and

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

Let $A = A_1 A_2 \cdots A_m = g_p^{\sum_{i \in [m]} \alpha_i}$

Next:

$$(\alpha_1 x_1 + \cdots + \alpha_m x_m)(\alpha_1 + \cdots + \alpha_m) = \sum_{i \in [m]} \alpha_i^2 x_i + \sum_{i \neq j} \alpha_i \alpha_j x_i$$

Same expressions modulo cross terms!

Consider the exponent:

$$(\alpha_1 x_1 + \cdots + \alpha_m x_m)^2 = \sum_{i \in [m]} \alpha_i^2 x_i^2 + \sum_{i \neq j} \alpha_i \alpha_j x_i x_j$$

cross-terms

Proving Relations on Committed Values

common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$

$$\forall i \neq j: B_{ij} = g_p^{\alpha_i \alpha_j}$$

Approach: augment CRS with cross-terms

$$A = g_p^{\alpha_1 + \cdots + \alpha_m}$$

commitment to $(x_1, \ldots, x_m)$

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

If $x_i^2 = x_i$ for all i, then these expressions are equal up to cross-terms

Prover now computes cross terms

$$V = \prod_{i \neq j} B_{i,j}^{x_i - x_i x_j} = g_p^{\sum_{i \neq j} \alpha_i \alpha_j x_i x_j - \alpha_i \alpha_j x_i}$$

Verifier now checks:

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

Next:

$$(\alpha_1 x_1 + \cdots + \alpha_m x_m)(\alpha_1 + \cdots + \alpha_m) = \sum_{i \in [m]} \alpha_i^2 x_i + \sum_{i \neq j} \alpha_i \alpha_j x_i$$

Same expressions modulo cross terms!

Consider the exponent:

$$(\alpha_1 x_1 + \cdots + \alpha_m x_m)^2 = \sum_{i \in [m]} \alpha_i^2 x_i^2 + \sum_{i \neq j} \alpha_i \alpha_j x_i x_j$$

cross-terms

Proving Relations on Committed Values

common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$

$$\forall i \neq j: B_{ij} = g_p^{\alpha_i \alpha_j}$$

Approach: augment CRS with cross-terms

$$A = g_p^{\alpha_1 + \cdots + \alpha_m}$$

commitment to $(x_1, \ldots, x_m)$

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

If $x_i^2 = x_i$ for all i, then these expressions are equal up to cross-terms

Prover now computes cross terms

$$V = \prod_{i \neq j} B_{i,j}^{x_i - x_i x_j} = g_p^{\sum_{i \neq j} \alpha_i \alpha_j x_i x_j - \alpha_i \alpha_j x_i}$$

Verifier now checks:

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

$$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e(g_p, g_p)^{\boxed{\sum_{i \in [m]} \alpha_i^2 x_i^2} + \sum_{i \neq j} \alpha_i \alpha_j x_i x_j}$$

$\|$ if $x_i = x_i^2$

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

$$e(g_p, V) = e(g_p, g_p)^{\sum_{i \neq j} \alpha_i \alpha_j x_i x_j - \alpha_i \alpha_j x_i}$$

Proving Relations on Committed Values

common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$

$$\forall i \neq j \colon B_{ij} = g_p^{\alpha_i \alpha_j}$$

Approach: augment CRS with cross-terms

$$A = g_p^{\alpha_1 + \cdots + \alpha_m}$$

commitment to $(x_1, \dots, x_m)$

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

If $x_i^2 = x_i$ for all i, then these expressions are equal up to cross-terms

Prover now computes cross terms

$$V = \prod_{i \neq j} B_{i,j}^{x_i - x_i x_j} = g_p^{\sum_{i \neq j} \alpha_i \alpha_j x_i x_j - \alpha_i \alpha_j x_i}$$

Verifier now checks:

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

$$e(\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) = e(g_p, g_p)^{\sum_{i \in [m]} \alpha_i^2 x_i^2 + \boxed{\sum_{i \neq j} \alpha_i \alpha_j x_i x_j}}$$

$$\Vert$$

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

$$+$$

$$e(g_p, V) = e(g_p, g_p)^{\sum_{i \neq j} \alpha_i \alpha_j x_i x_j - \alpha_i \alpha_j x_i}$$

Proving Relations on Committed Values

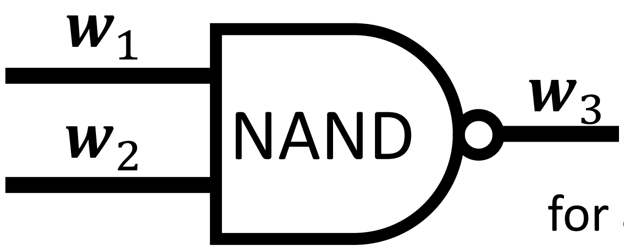
common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$
$$A = g_p^{\alpha_1 + \cdots + \alpha_m}$$

$$\forall i \neq j: B_{ij} = g_p^{\alpha_i \alpha_j}$$

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:

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

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

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

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!

Proving Relations on Committed Values

common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$
$$A = g_p^{\alpha_1 + \cdots + \alpha_m}$$

$$\forall i \neq j \colon B_{ij} = g_p^{\alpha_i \alpha_j}$$

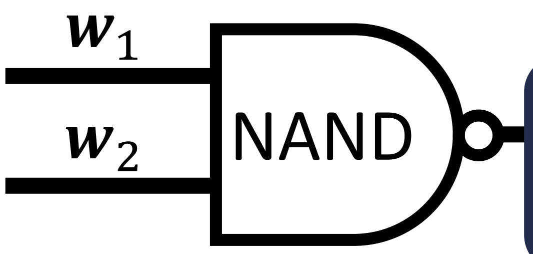
If $w_{3,i} + w_{1,i} w_{2,i} = 1$ for all i, then

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

only consists of cross terms!

Gate validity

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



Generalizes to arbitrary quadratic predicates

Can leverage same approach as before:

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

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

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

Is This Sound?

common reference string

$$A_1 = g_p^{\alpha_1}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$

$$\forall i \neq j: B_{ij} = g_p^{\alpha_i \alpha_j}$$

$$A = g_p^{\alpha_1 + \cdots + \alpha_m}$$

commitment to $(x_1, \ldots, x_m)$

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

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* – <u>cannot</u> extract a full witness

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

Approach: Program the CRS to extract a witness for instance i

Implies non-adaptive (and semi-adaptive) soundness

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

Fix any tuple $(x_1, \dots, x_m)$ where $x_i \notin \mathcal{L}_C$ for some i

Suppose prover constructs accepting proof π of $(x_1, \dots, x_m)$

Switch CRS to be extracting on i

CRS indistinguishability implies that proof still verifies

In extracting mode, we can recover w_i such that $C(x_i, w_i) = 1$ so $x_i \in \mathcal{L}_C$

Local Extraction

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

$$A = \prod_{i\in[m]} g_p^{\alpha_i}$$

$$B_{ij} = g_p^{\alpha_i \alpha_j} = A_i^{\alpha_j}$$

$$\forall i \neq j$$

Move slot i^* to full group

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

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

$$B_{ij} = A_i^{\alpha_j} \quad \forall i \neq j \neq i^*$$

$$B_{i^*j} = B_{ji^*} = A_{i^*}^{\alpha_j}$$

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 $\mathbb{G}_q$)

Consider a commitment $\sigma_{\boldsymbol{x}}$:

$$\sigma_{\boldsymbol{x}} = A_1^{x_1} A_2^{x_2} \cdots A_{i^*-1}^{x_{i^*-1}} A_{i^*}^{x_{i^*}} A_{i^*+1}^{x_{i^*+1}} \cdots A_m^{x_m}$$

$$= g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m} g_q^{r x_{i^*}}$$

Project into $\mathbb{G}_q$

Compute $z \leftarrow e(\sigma_{\boldsymbol{x}}, g_q)$

if $z = 1$, output $x_{i^*} = 0$

if $z \neq 1$, output $x_{i^*} = 1$

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 key 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 key 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 $\mathbb{G}$

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

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

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:

$$e(\sigma_{w_3}, A)e(\sigma_{w_1}, \sigma_{w_2}) = e(A, A)e(g_p, W)$$

Correctness of Extraction

Consider gate validity check:

$$e(\sigma_{w_3}, A)e(\sigma_{w_1}, \sigma_{w_2}) = e(A, A)e(g_p, W)$$

Adversary chooses commitment $\sigma_{w_1}, \sigma_{w_2}, \sigma_{w_3}$ 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:

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

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:

$$e(\sigma_{w_3}, A)e(\sigma_{w_1}, \sigma_{w_2}) = e(A, A)e(g_p, W)$$

Adversary chooses commitment $\sigma_{w_1}, \sigma_{w_2}, \sigma_{w_3}$ 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 $A = g_p^{\sum_{i \in [m]} \alpha_i} g_q^r$

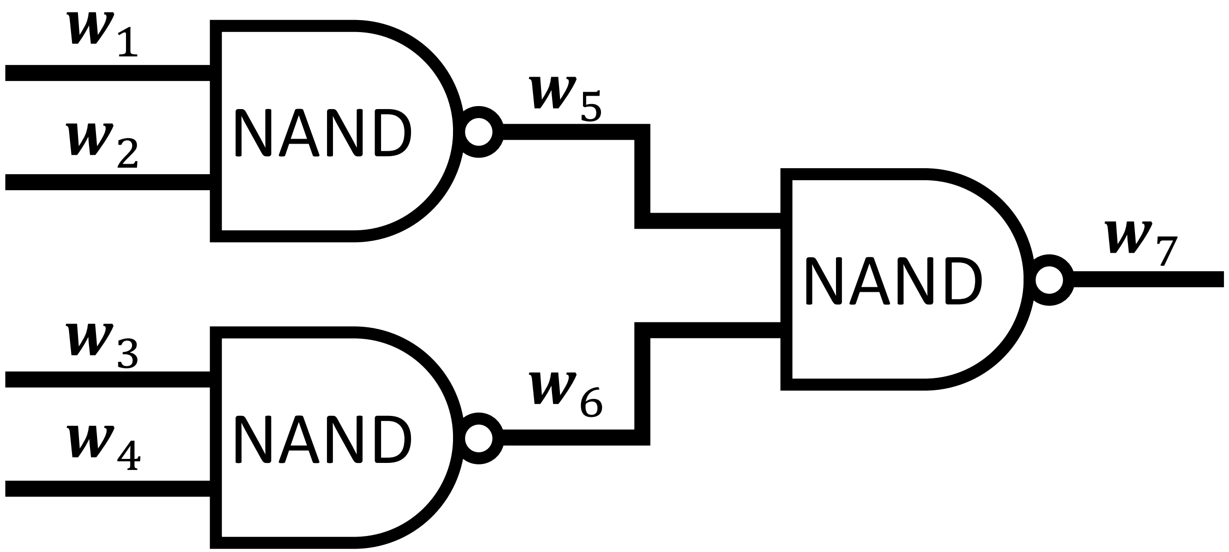
In the order-q subgroup, exponents must satisfy:

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

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 BARGs



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

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

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

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

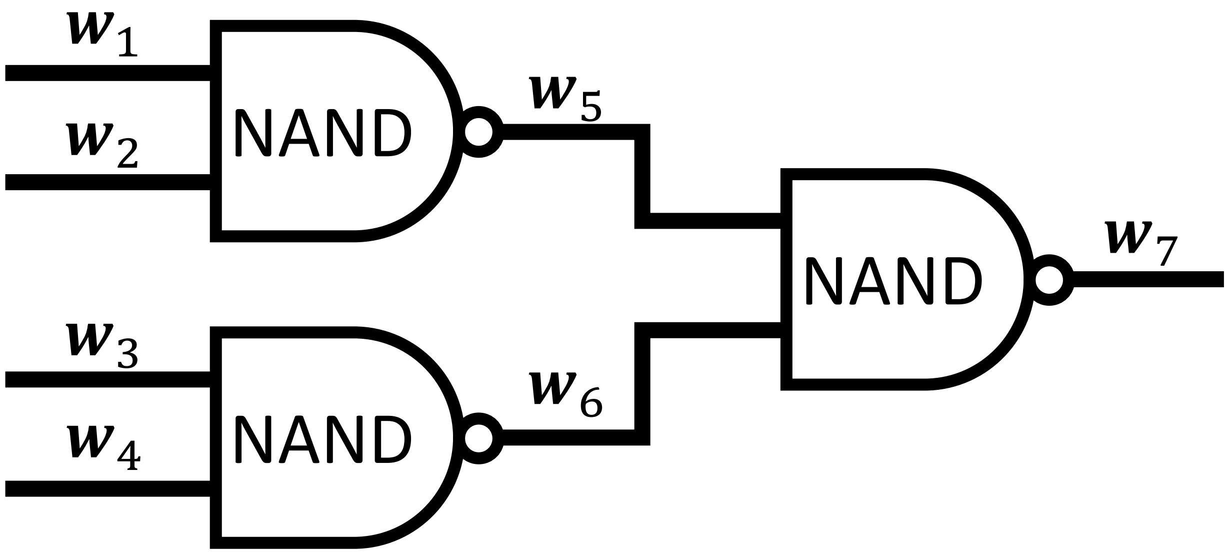
2 Prover constructs the following proofs:

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

Remaining checks ensure that statement correctly encoded and output is 1

Implication: Successful extraction of valid witness for instance i^*

Proof Size



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

1 Prover commits to each vector of wire assignments

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

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

Single group element

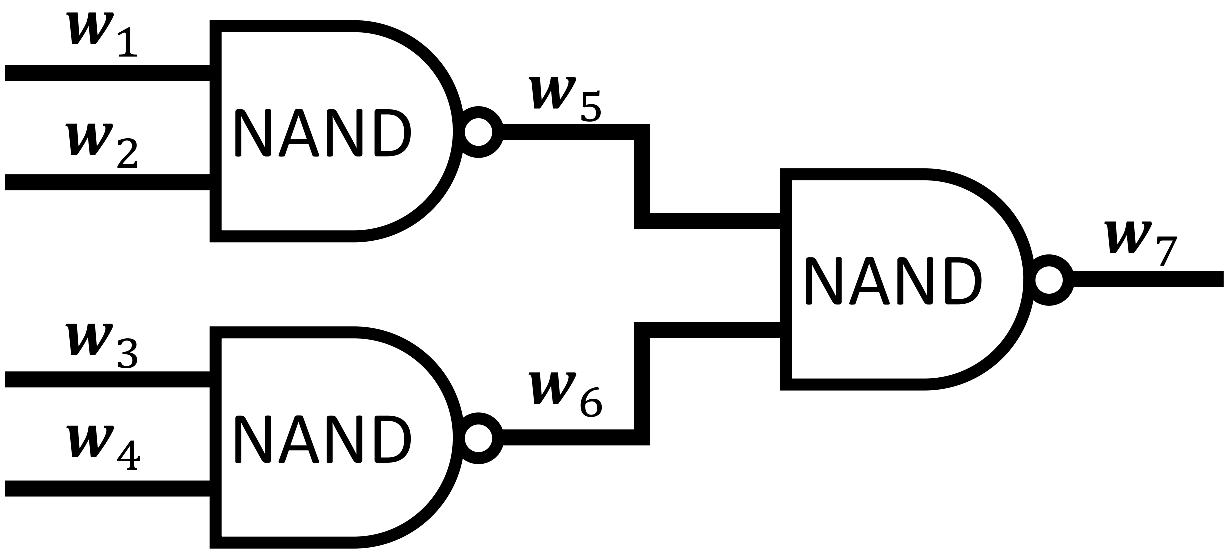
2 Prover constructs the following proofs:

Input validity	
Wire validity	One group element
Gate validity	One group element
Output validity	

Overall proof size (t wires, s gates):

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

Verification Time



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

1 Prover commits to each vector of wire assignments

2 Prover constructs the following proofs:

Input validity	$O(mn)$ group operations
Wire validity	$O(1)$ group operations
Gate validity	$O(1)$ group operations
Output validity	Equality check

Overall verification time:

$$nm \cdot \mathrm{poly}(\lambda) + |C| \cdot \mathrm{poly}(\lambda)$$

From Composite-Order to Prime-Order

BARGs 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

$$\langle g^{\alpha\boldsymbol{u}+\beta\boldsymbol{v}} \rangle \quad \langle g^{\boldsymbol{u}} \rangle \quad \langle g^{\boldsymbol{v}} \rangle$$

full space ($\mathbb{Z}_p^2$)

subspaces of $\mathbb{Z}_p^2$

Simulate subgroups with subspaces

prime-order group

$\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_p^2$ (linearly independent)

From Composite-Order to Prime-Order

BARGs 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

$$\langle g^{\alpha \boldsymbol{u} + \beta \boldsymbol{v}} \rangle \quad \langle g^{\boldsymbol{u}} \rangle \quad \langle g^{\boldsymbol{v}} \rangle$$

Simulate subgroups with subspaces

prime-order group

Normal mode: $g_p^{\alpha_i} \to g^{\alpha_i \boldsymbol{u}}$

Extracting scheme: $g_p^{\alpha_i} g_q^{r} \to g^{\alpha_i \boldsymbol{u} + r\boldsymbol{v}}$

Indistinguishable under DDH

From Composite-Order to Prime-Order

BARGs 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

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$\langle g^{\alpha \boldsymbol{u} + \beta \boldsymbol{v}} \rangle \quad \langle g^{\boldsymbol{u}} \rangle \quad \langle g^{\boldsymbol{v}} \rangle$$

Simulate subgroups with subspaces

prime-order group

Technically: move to asymmetric pairing-groups first (otherwise DDH does not hold)

Indistinguishable under DDH

From Composite-Order to Prime-Order

BARGs 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

$$\langle g^{\alpha \boldsymbol{u}+\beta \boldsymbol{v}} \rangle \quad \langle g^{\boldsymbol{u}} \rangle \quad \langle g^{\boldsymbol{v}} \rangle$$

prime-order group

Pairing is an outer product:

$$e(g^{\boldsymbol{u}}, g^{\boldsymbol{v}}) = e(g,g)^{\boldsymbol{u} \otimes \boldsymbol{v}} = e(g,g)^{\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}}$$

From Composite-Order to Prime-Order

BARGs for NP from standard assumptions over bilinear maps

Subgroup decision assumption in composite-order bilinear groups

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

$$\langle g^{\alpha\boldsymbol{u}+\beta\boldsymbol{v}}\rangle \quad \langle g^{\boldsymbol{u}}\rangle \quad \langle g^{\boldsymbol{v}}\rangle$$

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

Composite-order setting: $e(g_p, V)$ cannot contain a $\mathbb{G}_q$ component ⇒ isolate instance i^* in $\mathbb{G}_q$ subgroup

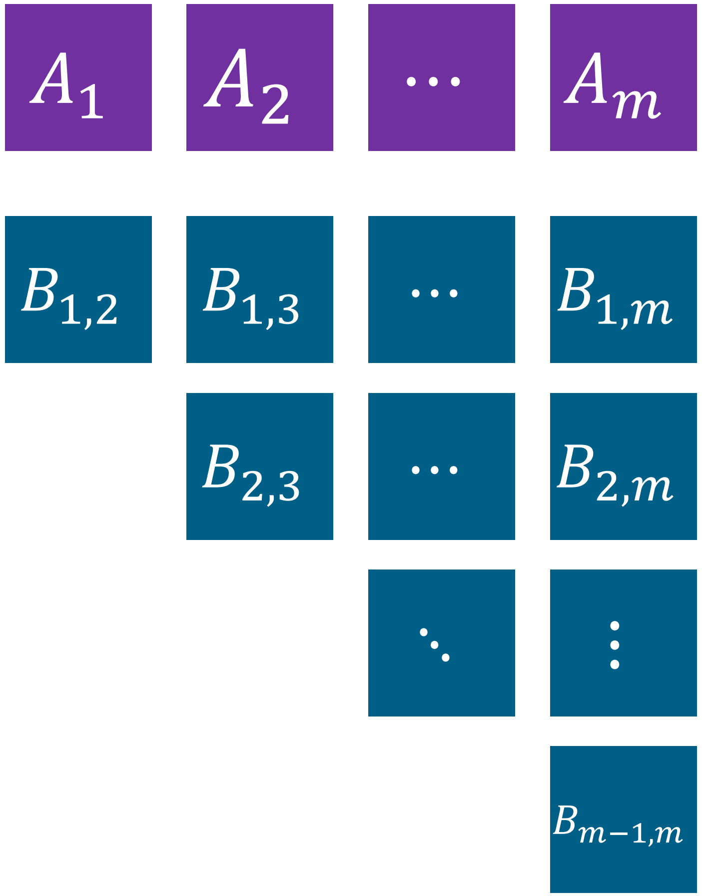
Prime-order setting: $e(g^{\boldsymbol{u}}, V)$ cannot contain a $\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}$ component ⇒ isolate instance i^* in $\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}$ subspace

Generalizes to yield a BARG 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 $m^2 \cdot \mathrm{poly}(\lambda)$

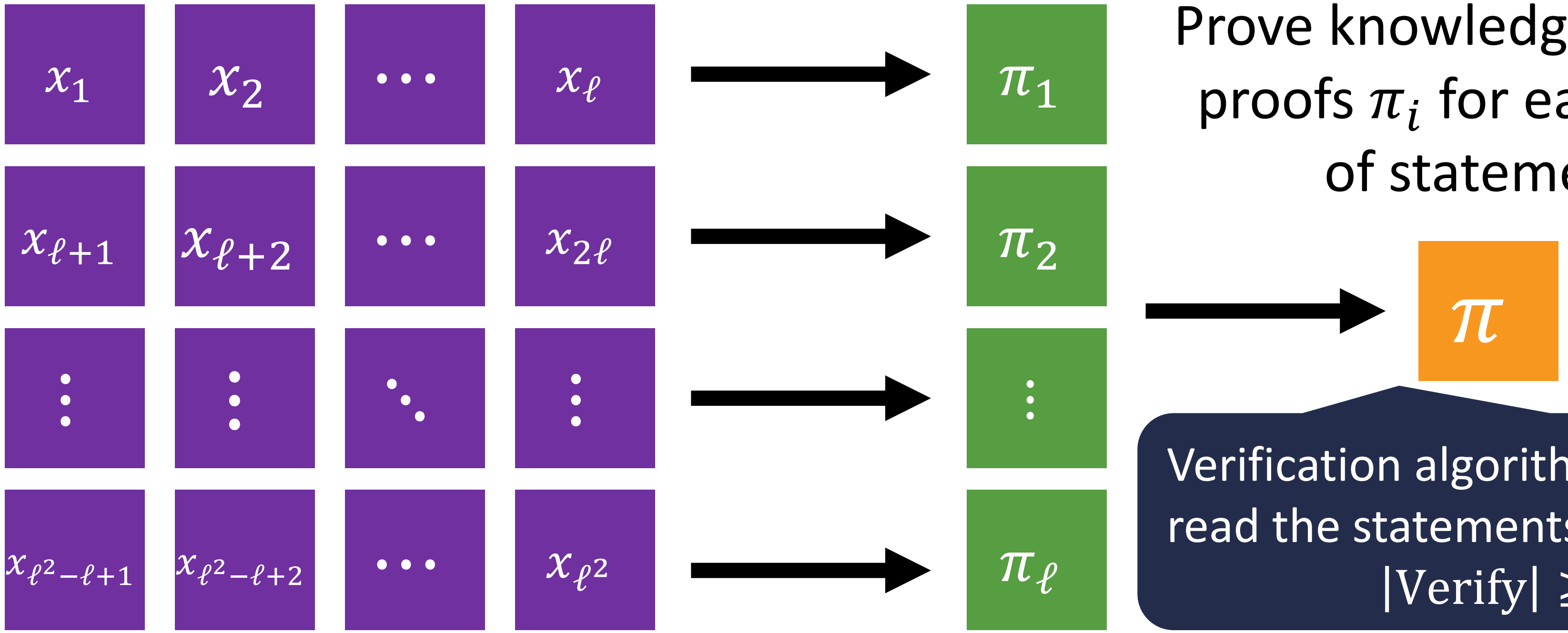
Can rely on recursive composition to reduce CRS size:

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

for any constant $\varepsilon > 0$

Similar approach as [KPY19]

The Base Case



Prove knowledge of BARG proofs 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 BARG on $\ell = \sqrt{m}$ instances to prove each batch

Soundness <u>necessitates</u> somewhere extractability of base BARG

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

BARGs with Split Verification

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

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

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

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

Preprocesses statements into a short 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

$$\begin{array}{cccc} x_1 & x_2 & \cdots & x_\ell \\ x_{\ell+1} & x_{\ell+2} & \cdots & x_{2\ell} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\ell^2-\ell+1} & x_{\ell^2-\ell+2} & \cdots & x_{\ell^2} \end{array} \quad \begin{array}{c} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{array} \quad \begin{array}{c} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_\ell \end{array} \longrightarrow \pi$$

Prove knowledge of BARG proofs π_i for each batch of statements

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

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

BARG used to check the relation

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

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

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

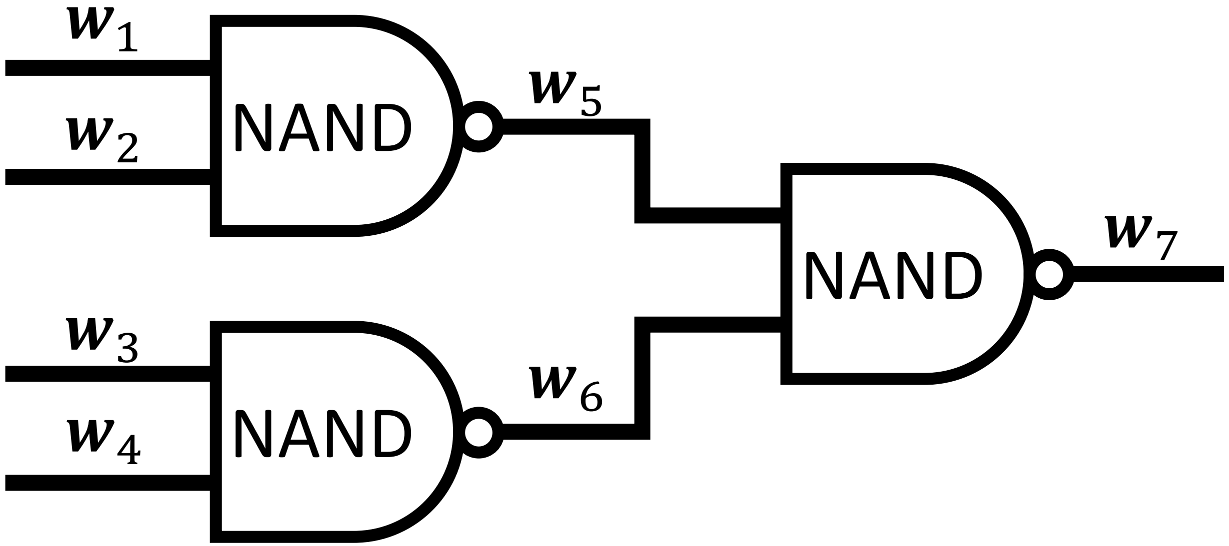
$$\ell = \sqrt{m}$$

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

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

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

BARG with Split Verification



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

(no more input validity checks)

Verifier checks the following

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

Wire validity, **Gate validity**, **Output validity** — $|C| \cdot \mathrm{poly}(\lambda)$

constant number of group operations per wire/gate

Only depends on the statement!

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

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

$$\mathrm{GenVK}\big(\mathrm{crs}, (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)\big) \rightarrow (\sigma_1, \ldots, \sigma_n)$$

BARGs with Short CRS

Corollary: BARGs 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

BARG 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:

succinct argument for polynomial-time computations

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

This work (from k-Lin)

succinct vector commitment that allows extracting on single index

Recall vector commitment we use for committing to wire values:

$$A_1, \ldots, A_m, \boldsymbol{x} \rightarrow A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m}$$

Same technique (cross-term cancellation) yields a somewhere extractable commitment (in combination with somewhere statistically binding hash functions [HW15])

Application to RAM Delegation ("SNARGs for P")

Choudhuri et al. [CJJ21] showed:

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

This work (from k-Lin)

This work + [OPWW15] (from SXDH)

Recall vector commitment we use for committing to wire values:

$$A_1, \ldots, A_m, \boldsymbol{x} \to A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m}$$

Same technique (cross-term cancellation) yields a somewhere extractable commitment (in combination with somewhere statistically binding hash functions [HW15])

Application to RAM Delegation (“SNARGs for P”)

Choudhuri et al. [CJJ21] showed:

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

This work (from k-Lin)

This work + [OPWW15] (from SXDH)

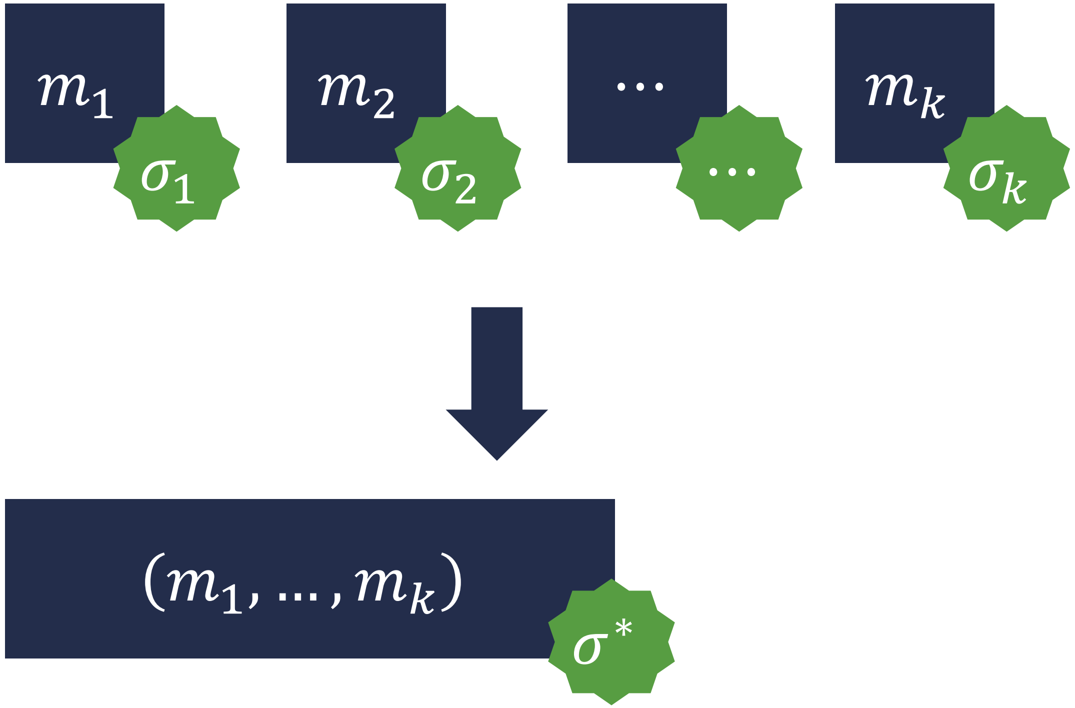
Corollary. RAM delegation from SXDH on prime-order pairing groups

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



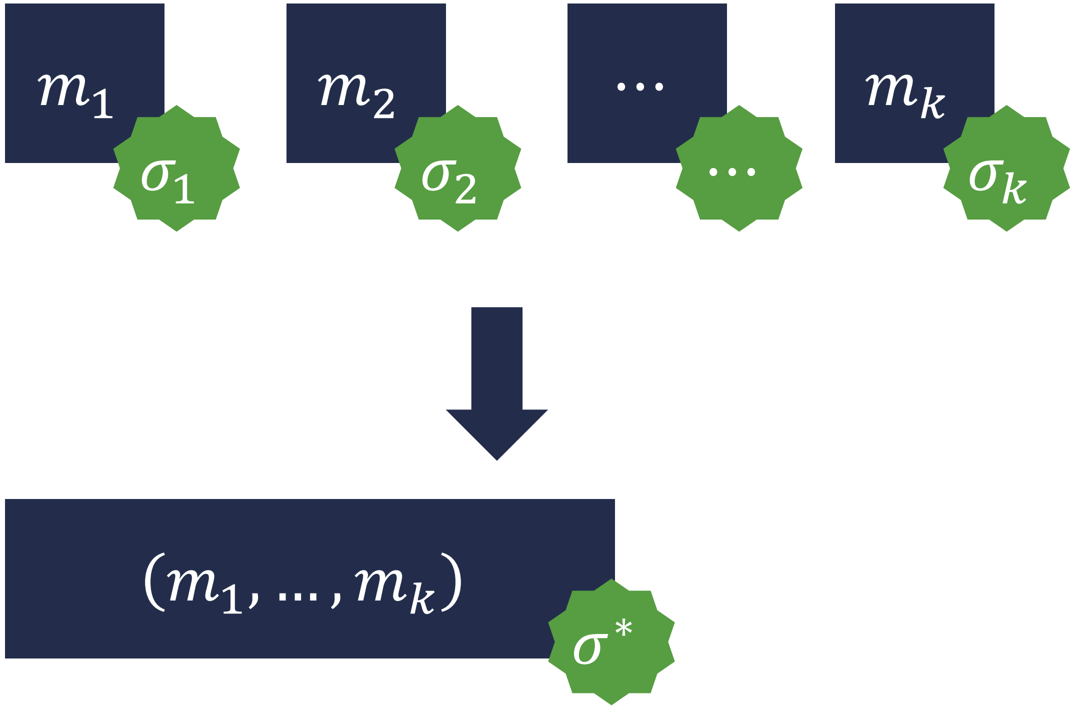
Given k message-signature pairs (m_i, σ_i)

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

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

prove knowledge of $\sigma_1, \ldots, \sigma_k$ 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 $(m_1, \ldots, m_k)$:
$|\sigma^*| = \text{poly}(\lambda, \log k)$

Can replace SNARKs for NP with a (somewhere extractable) BARG for NP:

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

Application to Aggregate Signatures

Can replace SNARKs for NP with a (somewhere extractable) BARG for NP:

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

This work: BARG 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

BARGs 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

Open Question: BARG with unbounded number of instances from bilinear maps

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

Thank you!