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 has \( m \) statements and wants to convince verifier that

\[ x_i \in \mathcal{L}_C \text{ for all } i \in [m] \]
Batch Arguments for NP

Boolean circuit satisfiability

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

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

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

\( \pi \) is the proof, \( (x_1, \ldots, x_m) \) is the input, and \( \lambda \) is the security parameter.

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

- \( \lambda \): security parameter
- Proof size can scale with circuit size (not a SNARG for NP)
Goal: Amortize the Cost of NP Verification

Boolean circuit satisfiability

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

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

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

In general setting, verifier needs to read statements
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]
Prover commits to each vector of wire assignments

\[ w_i = (w_{i,1}, ..., w_{i,m}) \]

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

**Our construction:** \( |\sigma_i| = \text{poly}(\lambda) \)
A Commit-and-Prove Strategy for BARGs

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

1. Prover commits to each vector of wire assignments

   \[
   \mathbf{w}_i = \begin{pmatrix} w_{i,1} & w_{i,2} & \cdots & w_{i,m} \end{pmatrix} \rightarrow \sigma_i
   \]

   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 $w_i = (w_{i,1}, ..., w_{i,m})$ be a vector of wire labels associated with wire $i$.

1. Prover commits to each vector of wire assignments

   $$w_i = \begin{bmatrix} w_{i,1} & w_{i,2} & \cdots & w_{i,m} \end{bmatrix}$$

   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**

Commitment for each wire is a commitment to a 0/1 vector.
A Commit-and-Prove Strategy for BARGs

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

1. Prover commits to each vector of wire assignments

\[
\mathbf{w}_i = [w_{i,1}, w_{i,2}, \ldots, w_{i,m}] \quad \rightarrow \quad \sigma_i
\]

**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 BARGs

Prover commits to each vector of wire assignments

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

1. Prover commits to each vector of wire assignments

$w_i = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,m} \end{bmatrix} \rightarrow \sigma_i$

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
- Output validity

Commitment to output wire is a commitment to the all-ones vector
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 $x = (x_1, \ldots, x_m) \in \{0,1\}^m$:

$$\sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m}$$
(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 \((x_1, \ldots, x_m)\)

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

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 \(G\):

\[ e(g^x, g^y) = e(g, g)^{xy} \]

\[ e(\sigma_x, \sigma_x) = e \left( g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}, g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m} \right) \]
\[ = e(g_p, g_p)^{(\alpha_1 x_1 + \cdots + \alpha_m x_m)^2} \]

Consider the exponent:

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

\begin{align*}
A_1 &= g_p^{\alpha_1} \\
A_2 &= g_p^{\alpha_2} \\
&\vdots \\
A_m &= g_p^{\alpha_m}
\end{align*}

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

\begin{align*}
\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}
\end{align*}

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 \(G\):

\[ e(g^x, g^y) = e(g, g)^{xy} \]

\[ e(\sigma_x, \sigma_x) = e\left(g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}, g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m}\right) = e(g_p, g_p)^{(\alpha_1 x_1 + \cdots + \alpha_m x_m)^2} \]

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 \((x_1, \ldots, x_m)\)

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

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:

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

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

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

\[A = g_p^{\alpha_1 + \cdots + \alpha_m}\]

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

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

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

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

Let \(A = A_1 A_2 \cdots A_m = g_p^{\Sigma_{i \in [m]} \alpha_i}\)

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:

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

Same expressions modulo cross-terms!
Proving Relations on Committed Values

common reference string

\[ A_1 = g_p^{\alpha_1} \quad \forall i \neq j: B_{ij} = g_p^{\alpha_i \alpha_j} \]
\[ A_2 = g_p^{\alpha_2} \quad \vdots \]
\[ A_m = g_p^{\alpha_m} \]

\[ A = g_p^{\alpha_1 + \ldots + \alpha_m} \]

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

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

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

Prover now computes cross terms

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

Verifier now checks:

\[ e(\sigma_x, \sigma_x) = e(\sigma_x, A)e(g_p, V) \]

Next:

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

Consider the exponent:

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

Same expressions modulo cross terms!
Proving Relations on Committed Values

common reference string

$$A_1 = g_p^{\alpha_1} \quad \forall i \neq j: B_{ij} = g_p^{\alpha_i \alpha_j}$$
$$A_2 = g_p^{\alpha_2}$$
$$\vdots$$
$$A_m = g_p^{\alpha_m}$$

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

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

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

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

Prover now computes cross terms

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

Verifier now checks:

$$e(\sigma_x, \sigma_x) = e(\sigma_x, A)e(g_p, V)$$

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

if $x_i = x_i^2$

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

$$e(g_p, V) = e(g_p, g_p) \Sigma_{i \neq j} \alpha_i \alpha_j x_i x_j - \alpha_i \alpha_j x_i$$
Proving Relations on Committed Values

**common reference string**

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

\[ A = g_p^{\alpha_1 + \cdots + \alpha_m} \]

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

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

**Approach:** augment CRS with cross-terms

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

**Prover now computes cross terms**

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

Verifier now checks:

\[ e(\sigma_x, \sigma_x) = e(\sigma_x, A) e(g_p, V) \]

\[ e(\sigma_x, \sigma_x) = e(g_p, g_p)^{\sum_{i \in [m]} \alpha_i^2 x_i^2 + \sum_{i \neq j} \alpha_i \alpha_j x_i x_j} \]
\[ e(\sigma_x, A) = e(g_p, g_p)^{\sum_{i \in [m]} \alpha_i^2 x_i} + e(g_p, V) \]
\[ e(g_p, V) = e(g_p, g_p)^{\sum_{i \neq j} \alpha_i \alpha_j x_i x_j - \alpha_i \alpha_j x_i} \]
Proving Relations on Committed Values

common reference string

\[ A_1 = g_p^{\alpha_1}, \quad A_2 = g_p^{\alpha_2}, \quad \ldots, \quad A_m = g_p^{\alpha_m} \]
\[ A = g_p^{\alpha_1 + \ldots + \alpha_m} \]

Gate validity

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

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

Can leverage same approach as before:

\[ e(\sigma_{w_3}, A) = e(g_p, g_p) \left( \sum_{i \in [m]} \alpha_i^2 w_{3,i} + \sum_{i \neq j} \alpha_i \alpha_j w_{3,i} \right) \]
\[ e(A, A) = e(g_p, g_p) \left( \sum_{i \in [m]} \alpha_i^2 + \sum_{i \neq j} \alpha_i \alpha_j \right) \]
\[ e(\sigma_{w_1}, \sigma_{w_2}) = e(g_p, g_p) \left( \sum_{i \in [m]} \alpha_i^2 w_{1,i} w_{2,i} + \sum_{i \neq j} \alpha_i \alpha_j w_{1,i} w_{2,j} \right) \]

If \( w_{3,i} + w_{1,i} w_{2,i} = 1 \) for all \( i \), then

\[ e(\sigma_{w_3}, A) e(\sigma_{w_1}, \sigma_{w_2}) e(A, A) \]

only consists of cross terms!
Proving Relations on Committed Values

common reference string

\[ A_1 = g_p^{\alpha_1} \]
\[ A_2 = g_p^{\alpha_2} \quad \forall i \neq j: B_{ij} = g_p^{\alpha_i \alpha_j} \]
\[ \vdots \]
\[ A_m = g_p^{\alpha_m} \]
\[ A = g_p^{\alpha_1 + \ldots + \alpha_m} \]

Gate validity

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

\[ e(\sigma_{w_3}, A) = e(g_p, g_p)^{\sum_{i \in [m]} \alpha_i^2 w_{3,i} + \sum_{i \neq j} \alpha_i \alpha_j w_{3,i}} \]
\[ e(A, A) = e(g_p, g_p)^{\sum_{i \in [m]} \alpha_i^2 + \sum_{i \neq j} \alpha_i \alpha_j} \]
\[ e(\sigma_{w_1}, \sigma_{w_2}) = e(g_p, g_p)^{\sum_{i \in [m]} \alpha_i^2 w_{1,i} w_{2,i} + \sum_{i \neq j} \alpha_i \alpha_j w_{1,i} w_{2,j}} \]
common reference string

\[ A_1 = g_p^{\alpha_1} \]
\[ A_2 = g_p^{\alpha_2} \]
\[ \vdots \]
\[ A_m = g_p^{\alpha_m} \]
\[ A = g_p^{\alpha_1 + \ldots + \alpha_m} \]

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

\[ \sigma_x = A_1^{x_1} A_2^{x_2} \ldots A_m^{x_m} \]
\[ = g_p^{\alpha_1 x_1 + \ldots + \alpha_m x_m} \]

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” \( [CJJ21] \)

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

Implies non-adaptive (and semi-adaptive) soundness
CRS will have two modes:

- **Normal mode**: used in the real scheme
- **Extracting on index $i$**: supports witness extraction for instance $i$ (given a trapdoor)

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 $(x_1, \ldots, x_m)$ where $x_i \notin \mathcal{L}_C$ for some $i$
- Suppose prover constructs accepting proof $\pi$ of $(x_1, \ldots, x_m)$
- Switch CRS to be extracting on $i$
- 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

Normal mode:

\[ A = \prod_{i \in [m]} g_{p}^{\alpha_i} \]

\[ B_{ij} = g_{p}^{\alpha_i \alpha_j} = A_i^{\alpha_j} \quad \forall i \neq j \]

Move slot \( i^* \) to full group

Extracting mode: (extract on \( i^* \))

\[ A = g_{q}^{r} \prod_{i \in [m]} g_{p}^{\alpha_i} \]

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

\[ B_{i^*j} = B_{ji^*} = A_i^{\alpha_j} \]

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 $i^*$):

Consider a commitment $\sigma_x$:

\[
\sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_{i^*-1}^{x_{i^*-1}} A_{i^*}^{x_{i^*}} A_{i^*+1}^{x_{i^*+1}} \cdots A_m^{x_m}
\]

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

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

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

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

Consider wire validity check:

\[ e(\sigma_x, \sigma_x) = e(\sigma_x, A)e(g_p, V) \]
Correctness of Extraction

Consider wire validity check:

\[ e(\sigma_x, \sigma_x) = e(\sigma_x, A)e(g_p, V) \]

Adversary chooses commitment \( \sigma_x \) and proof \( V \)
Correctness of Extraction

Consider wire validity check:

\[ e(\sigma_x, \sigma_x) = e(\sigma_x, A)e(g_p, V) \]

Adversary chooses commitment \( \sigma_x \) 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 \( \sigma_x = g_p^s g_q^t \)

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

In the order-\( q \) subgroup, exponents must satisfy:

\[ t^2 = tr \mod q \]
Consider wire validity check:

\[ e(\sigma_x, \sigma_x) = e(\sigma_x, A)e(g_p, V) \]

Adversary chooses commitment \( \sigma_x \) 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:** \( e(g_p, V) \) is always in the order-\( p \) subgroup; adversary cannot influence the verification relation in the order-\( q \) subgroup.

Write \( \sigma_x = g_p^s g_q^t \)

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

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

In the order-\( q \) subgroup, exponents must satisfy:

\[ t^2 = tr \mod q \]
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

\[
\begin{align*}
\sigma_{w_1} &= g_p^{s_1} g_q^{t_1} \\
\sigma_{w_2} &= g_p^{s_2} g_q^{t_2} \\
\sigma_{w_3} &= g_p^{s_3} g_q^{t_3}
\end{align*}
\]

In the order-\(q\) subgroup, exponents must satisfy:

\[ t_3 r + t_1 t_2 = r^2 \mod q \]

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

\[ b_3 r^2 + b_1 b_2 r^2 = r^2 \mod q \]

\[ b_3 = 1 - b_1 b_2 = \text{NAND}(b_1, b_2) \]
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

\[
\begin{align*}
\sigma_{w_1} &= g_p^{s_1} g_q^{t_1} \\
\sigma_{w_2} &= g_p^{s_2} g_q^{t_2} \\
\sigma_{w_3} &= g_p^{s_3} g_q^{t_3}
\end{align*}
\]

In the order-\( q \) subgroup, exponents must satisfy:

\[ t_3 r + t_1 t_2 = r^2 \mod q \]

Conclusion: extracted bits are consistent with gate operation

\[ b_3 = 1 - b_1 b_2 = \text{NAND}(b_1, b_2) \]
A Commit-and-Prove Strategy for BARGs

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

1. Prover commits to each vector of wire assignments:
   \[
   \mathbf{w}_i = [w_{i,1}, w_{i,2}, \ldots, w_{i,m}] \quad \Rightarrow \quad \sigma_i
   \]

   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
   - 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 $\mathbf{w}_i = (w_{i,1}, \ldots, w_{i,m})$ be a vector of wire labels associated with wire $i$.

1. **Prover commits to each vector of wire assignments**
   
   $$\mathbf{w}_i = w_{i,1}, w_{i,2}, \ldots, w_{i,m} \xrightarrow{\sigma_i}$$

   **Commitment size:** $|\sigma_i| = \text{poly}(\lambda)$
   
   Single group element

2. **Prover constructs the following proofs:**
   
   - Input validity
   - Wire validity
   - Gate validity
   - Output validity

   **Overall proof size** ($t$ wires, $s$ gates):
   
   $$2t + s \cdot \text{poly}(\lambda) = |C| \cdot \text{poly}(\lambda)$$
Let $\mathbf{w}_i = (w_{i,1}, \ldots, w_{i,m})$ be vector of wire labels associated with wire $i$.

1. Prover commits to each vector of wire assignments.

$\mathbf{w}_i = [w_{i,1}, w_{i,2}, \ldots, w_{i,m}] \rightarrow \sigma_i$

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:

$$nm \cdot \text{poly}(\lambda) + |C| \cdot \text{poly}(\lambda)$$
From Composite-Order to Prime-Order

BARGs for NP from standard assumptions over bilinear maps

Subgroup decision assumption in composite-order bilinear groups

\[ \mathbb{G} \cong \mathbb{G}_p \times \mathbb{G}_q \]

Simulate subgroups with subspaces

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

subspaces of \( \mathbb{Z}_p^2 \)

prime-order group

\[ u, v \in \mathbb{Z}_p^2 \text{ (linearly independent)} \]
From Composite-Order to Prime-Order

BARGs for NP from standard assumptions over bilinear maps

Subgroup decision assumption in \textit{composite-order} bilinear groups

\[ G \cong G_p \times G_q \]

\[ \langle g^{\alpha u + \beta v} \rangle \rightarrow \langle g^u \rangle \rightarrow \langle g^v \rangle \]

Simulate \textit{subgroups} with \textit{subspaces}

Normal mode: \[ g_p^{\alpha_i} \rightarrow g^{\alpha_i u} \]

Extracting scheme: \[ g_p^{\alpha_i} g_q^r \rightarrow g^{\alpha_i u + rv} \]

Indistinguishable under DDH
BARGs for NP from standard assumptions over bilinear maps

Subgroup decision assumption in composite-order bilinear groups

\[ G \cong G_p \times G_q \]

Simulate subgroups with subspaces

\[ \langle g^{\alpha u + \beta v} \rangle, \langle g^u \rangle, \langle g^v \rangle \]

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

Indistinguishable under DDH
BARGs for NP from standard assumptions over bilinear maps

Subgroup decision assumption in composite-order bilinear groups

\[ \mathbb{G} \cong \mathbb{G}_p \times \mathbb{G}_q \]

Simulate subgroups with subspaces

Prime-order group

Pairing is an outer product:

\[ e(g^u, g^v) = e(g, g)^{u \otimes v} = e(g, g)^{uv^T} \]
From Composite-Order to Prime-Order

BARGs for NP from standard assumptions over bilinear maps

Subgroup decision assumption in composite-order bilinear groups

\[ G \cong G_p \times G_q \]

\[ \langle g^{\alpha u + \beta v} \rangle \quad \langle g^u \rangle \quad \langle g^v \rangle \]

\[ e(\sigma_x, \sigma_x) = e(\sigma_x, A)e(g_p, V) \]

**Composite-order setting:** \( e(g_p, V) \) cannot contain a \( G_q \) component \( \Rightarrow \) isolate instance \( i^* \) in \( G_q \) subgroup

**Prime-order setting:** \( e(g^u, V) \) cannot contain a \( vv^T \) component \( \Rightarrow \) isolate instance \( i^* \) in \( vv^T \) 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:

\[ A_1 \quad A_2 \quad \ldots \quad A_m \]

\[ B_{1,2} \quad B_{1,3} \quad \ldots \quad B_{1,m} \]

\[ B_{2,3} \quad \ldots \quad B_{2,m} \]

\[ \vdots \]

\[ B_{m-1,m} \]

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

Can rely on recursive composition to reduce CRS size:

\[ m^2 \cdot \text{poly}(\lambda) \rightarrow m^\varepsilon \cdot \text{poly}(\lambda) \]

for any constant \( \varepsilon > 0 \)

Similar approach as [KPY19]
The Base Case

\[ \pi \]

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

Soundness necessitates somewhere extractability of base BARG

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

Prove knowledge of BARG proofs \( \pi_i \) for each batch of statements

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

\[ |Verify| \geq \sqrt{m} \cdot \text{poly}(\lambda) \]
BARGs with Split Verification

\[ \text{Verify}(\text{crs}, C, (x_1, \ldots, x_m), \pi) \]

\[ \text{GenVK}(\text{crs}, (x_1, \ldots, x_m)) \rightarrow \text{vk} \]

- Runs in time \( \text{poly}(\lambda, m, n) \)
- \( |\text{vk}| = \text{poly}(\lambda, \log m, n) \)

\[ \text{OnlineVerify}(\text{vk}, C, \pi) \]

- Runs in time \( \text{poly}(\lambda, \log m, |C|) \)

Preprocesses statements into a short verification key

Fast online verification

(Similar property from [CJJ21])
Recursive Bootstrapping

Prove knowledge of BARG proofs $\pi_i$ 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

$R((C, vk_1, \ldots, vk_\ell), (\pi_1, \ldots, \pi_\ell)) = 1$

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

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

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

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

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

$\ell = \sqrt{m}$
Verifier checks the following:

- **Input validity**\( \{ w_1, \ldots, w_7 \} \)
- **Wire validity** \(|C| \cdot \text{poly}(\lambda)\)
- **Gate validity** constant number of group operations per wire/gate
- **Output validity**

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

(no more input validity checks)

Only depends on the statement!

Given \((x_1, \ldots, x_m) \in \{(0,1)^n\}^m\), verifier computes commitments to bits of the statement

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

\[
\text{GenVK}(\text{crs}, (x_1, \ldots, x_m)) \rightarrow (\sigma_1, \ldots, \sigma_n)
\]
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:** $|\text{crs}| = m^\epsilon \cdot \text{poly}(\lambda)$ for any constant $\epsilon > 0$
- **Proof size:** $|\pi| = \text{poly}(\lambda, |C|)$
- **Verification time:** $|\text{Verify}| = \text{poly}(\lambda, n, m) + \text{poly}(\lambda, |C|)$
Application to RAM Delegation ("SNARGs for P")

Choudhuri et al. [CJJ21] showed:

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

succinct vector commitment that allows extracting on single index

succinct argument for polynomial-time computations
Application to RAM Delegation ("SNARGs for P")

Choudhuri et al. [CJJ21] showed:

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

**succinct argument for polynomial-time computations**

Recall vector commitment we use for committing to wire values:

\[ A_1, \ldots, A_m, x \rightarrow A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m} \]

Same technique (cross-term cancellation) yields a somewhere extractable commitment (in combination with somewhere statistically binding hash functions [HW15])
Choudhuri et al. [CJJ21] showed:

BARG with split verification

\[ A_1, \ldots, A_m, x \rightarrow A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m} \]

Delegation scheme for RAM programs

Recall vector commitment we use for committing to wire values:

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

**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

Folklore construction from succinct arguments for NP (SNARKs for NP):
prove knowledge of $\sigma_1, \ldots, \sigma_k$ such that $\text{Verify}(vk, m_i, \sigma_i) = 1$

Given $k$ message-signature pairs $(m_i, \sigma_i)$:

Short signature $\sigma^*$ 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 $\text{Verify}(vk, m_i, \sigma_i) = 1$

Given $k$ message-signature pairs $(m_i, \sigma_i)$:

Short signature $\sigma^*$ 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 $\text{Verify}(vk, m_i, \sigma_i) = 1$

**This work:** BARG for **bounded** number of instances

**Corollary.** Aggregate signature supporting **bounded** 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]
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!