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

\[\pi = (w_1, \ldots, w_m)\]

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

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

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

“Proof size for a single instance”

\( \lambda \) : security parameter

Proof size scales \textit{sublinearly} with the number of instances
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: \(|\pi| = |C| \cdot \text{poly}(\log m, \lambda)\)

Similar* requirement on verification time

*Verifier does need to read statements so we do allow a poly(\(\lambda, m, n\)) dependence
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 $\mathbf{w}_i = (w_{i,1}, \ldots, w_{i,m})$ be a vector of wire labels associated with wire $i$ across the $m$ instances.

1. Prover commits to each vector of wire assignments.

Let $\mathbf{w}_i = (w_{i,1}, \ldots, w_{i,m})$

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

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

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

1. Prover commits to each vector of wire assignments
   \[
   \mathbf{w}_i = \begin{bmatrix} w_{i,1} & w_{i,2} & \cdots & w_{i,m} \end{bmatrix}
   \rightarrow \sigma_i
   \]

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

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

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

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

Prover commits to each vector of wire assignments:

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

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

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

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

1. Prover commits to each vector of wire assignments

   $w_i = w_{i,1} \; w_{i,2} \; ... \; w_{i,m} \; \sigma_i$

   **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 $w_i = (w_{i,1}, \ldots, w_{i,m})$ be a vector of wire labels associated with wire $i$ across the $m$ instances.

1. Prover commits to each vector of wire assignments:
   \[ w_i = \begin{array}{c} w_{i,1} \ w_{i,2} \ \cdots \ w_{i,m} \end{array} \rightarrow \sigma_i \]

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

1. Prover commits to each vector of wire assignments $w_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,m})$.

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

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

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

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 $\alpha_1, ..., \alpha_m \leftarrow \mathbb{Z}_N$
output $A_1 \leftarrow g_p^{\alpha_1}, ..., A_m \leftarrow g_p^{\alpha_m}$

denotes encodings in $\mathbb{G}_p$

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

$$\sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m}$$

(subset product of the $A_i$'s)

$$[\sigma_x] = [\sum_{i\in[m]} \alpha_i x_i]$$
Proving Relations on Committed Values

common reference string

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

\[ [\vdots] \]

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

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

\[ [\sum_{i \in [m]} \alpha_i x_i] \]

\[ \sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m} \]

\[ = g_p^{\alpha_1 x_1 + \cdots + \alpha_m x_m} \]

**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 \(\mathbb{G}\):

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

\[ e([x], [y]) \rightarrow [xy] \]

*Multiplies exponents in the target group*
Proving Relations on Committed Values

common reference string

\[
\begin{align*}
[\alpha_1] & \quad A_1 = g_p^{\alpha_1} \\
[\vdots] & \\
[\alpha_m] & \quad A_m = g_p^{\alpha_m}
\end{align*}
\]

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

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

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

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:

\[
e(\sigma_x, \sigma_x) \quad \text{and} \quad e(\sigma_x, \prod_{i \in [m]} A_i)
\]

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

(commitment to all-ones vector)
Proving Relations on Committed Values

**common reference string**

\[
\begin{align*}
[\alpha_1] & \quad A_1 = g_{p_1}^{\alpha_1} \\
[\vdots] & \\
[\alpha_m] & \quad A_m = g_{p_m}^{\alpha_m}
\end{align*}
\]

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

\[
[\sum_{i \in [m]} \alpha_i x_i]
\]

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

**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:

\[
e(\sigma_x, \sigma_x) \text{ and } e(\sigma_x, \prod_{i \in [m]} A_i)
\]

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

\[
= [\sum_{i \in [m]} \alpha_i^2 x_i^2] \times [\sum_{i \neq j} \alpha_i \alpha_j x_i x_j]
\]

**non-cross terms**

**cross terms**
Proving Relations on Committed Values

common reference string
\[ \begin{align*}
[\alpha_1] & \quad A_1 = g_p^{\alpha_1} \\
[\vdots] & \quad A_m = g_p^{\alpha_m}
\end{align*} \]

**Wire validity**
Commitment for each wire is a commitment to a 0/1 vector
\[ x \in \{0,1\} \text{ if and only if } x^2 = x \]

**Approach:** consider the following pairing relations:
\[ e(\sigma_x, \sigma_x) \text{ and } e(\sigma_x, \prod_{i\in[m]} A_i) \]

\[
\begin{align*}
e\left(\sum_{i\in[m]} \alpha_i x_i, \sum_{i\in[m]} \alpha_i \right) & \quad = \quad \left(\sum_{i\in[m]} \alpha_i^2 x_i \right) \times \left(\sum_{i\neq j} \alpha_i \alpha_j x_i \right) \\
& \quad \text{(non-cross terms)} \quad \text{(cross terms)}
\end{align*}
\]
\[
\begin{align*}
e\left(\sum_{i\in[m]} \alpha_i x_i, \sum_{i\in[m]} \alpha_i \right) & \quad = \quad \left(\sum_{i\in[m]} \alpha_i^2 x_i^2 \right) \times \left(\sum_{i\neq j} \alpha_i \alpha_j x_i x_j \right) \\
& \quad \text{(non-cross terms)} \quad \text{(cross terms)}
\end{align*}
\]
Proving Relations on Committed Values

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

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

$$= \left[ \sum_{i \in [m]} \alpha_i x_i \right]$$

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:

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

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

$$\equiv \left[ \sum_{i \in [m]} \alpha_i^2 x_i \right] \times \left[ \sum_{i \neq j} \alpha_i \alpha_j x_i \right]$$

non-cross terms

cross terms

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

$$\equiv \left[ \sum_{i \in [m]} \alpha_i^2 x_i^2 \right] \times \left[ \sum_{i \neq j} \alpha_i \alpha_j x_i x_j \right]$$

non-cross terms

cross terms
Proving Relations on Committed Values

If \( x_i^2 = x_i \) for all \( i \), then
\[
\left[ \sum_{i \in [m]} \alpha_i^2 x_i \right] = \left[ \sum_{i \in [m]} \alpha_i^2 x_i^2 \right]
\]

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:
\[
e(\sigma_x, \sigma_x) \text{ and } e(\sigma_x, \Pi_{i \in [m]} A_i)
\]

When \( x_i^2 = x_i \), difference between these terms is
\[
\left[ \sum_{i \neq j} \alpha_i \alpha_j (x_i - x_i x_j) \right]
\]

Give prover ability to eliminate cross-terms only

Augment CRS with cross-terms
\[
\left[ \alpha_i \alpha_j \right] B_{i,j} = g_p^{\alpha_i \alpha_j} \quad \forall \ i \neq j
\]
Proving Relations on Committed Values

Prover now computes additional group component in the base group

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

Pair with \( g_p \)

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

\[
e(g_p, V)
\]

Pair with \( g_p \)

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

Augment CRS with cross-terms

\[
\left[ \alpha_i \alpha_j \right]
\]

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

When \( x_i^2 = x_i \), difference between these terms is

\[
\left[ \sum_{i \in [m]} \alpha_i x_i \right]
\]

\[
\left[ \sum_{i \in [m]} \alpha_i \right]
\]

Give prover ability to eliminate cross-terms only
Proving Relations on Committed Values

Prover now computes additional group component in the base group

\[ [\sum_{i \neq j} \alpha_i \alpha_j (x_i - x_ix_j)] \]

Pair with \( g_p \)

\[ [\sum_{i \neq j} \alpha_i \alpha_j (x_i - x_ix_j)] \]

\( V = B_{i,j}^{x_i-x_ix_j} \)

\[ e(g_p, V) \]

Overall verification relation:

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

\[ A = \prod_{i \in [m]} A_i \]
Proving Relations on Committed Values

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

\[
\left[ \sum_{i \neq j} \alpha_i \alpha_j (x_i - x_ix_j) \right]
\]

Pair with \( g_p \)

\[
\left[ \sum_{i \neq j} \alpha_i \alpha_j (x_i - x_ix_j) \right] e(g_p, V)
\]

Overall verification relation:

\[
e(\sigma_x, \sigma_x) = e(\sigma_x, A)e(g_p, V) \quad A = \prod_{i \in [m]} A_i
\]

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

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

Pair with \( g_p \)

\[ e(g_p, V) \]

Overall verification relation:
\[ e(\sigma_x, \sigma_x) = e(\sigma_x, A) e(g_p, V) \]
\[ A = \prod_{i \in [m]} A_i \]

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:

\[
\begin{align*}
[\alpha_1] & \quad [\ldots] & \quad [\alpha_m] \\
A_1 &= g_p^{\alpha_1} & A_m &= g_p^{\alpha_m} \\
[\alpha_1 + \ldots + \alpha_m] & \quad A = \prod_{i \in [m]} A_i \\
[\alpha_i \alpha_j] & \quad B_{i,j} = g_p^{\alpha_i \alpha_j} \forall i \neq j
\end{align*}
\]

Gate validity

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

\[
\text{NAND} \quad w_3 = 1 - w_1, w_2
\]

for all \(i \in [m]\):

Relation is \textbf{quadratic} in the inputs

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

\[
\sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m}
\]

\[
= g_p^{\alpha_1 x_1 + \ldots + \alpha_m x_m}
\]

Can leverage similar approach as before
Proof Size

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:

$$w_i = [w_{i,1}, w_{i,2}, \ldots, w_{i,m}]$$

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):

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

Common reference string:

\[
\begin{align*}
[\alpha_1] & = g_p^{\alpha_1} \\
[\alpha_m] & = g_p^{\alpha_m} \\
[\alpha_1 + \cdots + \alpha_m] & = \prod_{i \in [m]} A_i \\
[\alpha_i \alpha_j] & = g_p^{\alpha_i \alpha_j} \forall i \neq j
\end{align*}
\]

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

\[
\left[ \sum_{i \in [m]} \alpha_i x_i \right]
\]

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

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]
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**
Local Extraction

Normal mode:

\[
\begin{align*}
A_1 & \quad g_p^{\alpha_1} \\
\cdots & \quad \cdots \\
A_{i^*-1} & \quad g_p^{\alpha_{i^*-1}} \\
A_{i^*} & \quad g_p^{\alpha_{i^*}} \\
A_{i^*+1} & \quad g_p^{\alpha_{i^*+1}} \\
\cdots & \quad \cdots \\
A_m & \quad g_p^{\alpha_m}
\end{align*}
\]

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

Extracting mode:

(\text{extract on } i^*)

\[
\begin{align*}
A_1 & \quad g_p^{\alpha_1} \\
\cdots & \quad \cdots \\
A_{i^*-1} & \quad g_p^{\alpha_{i^*-1}} \\
A_{i^*} & \quad g_p^{\alpha_{i^*}} g_q^r \\
A_{i^*+1} & \quad g_p^{\alpha_{i^*+1}} \\
\cdots & \quad \cdots \\
A_m & \quad g_p^{\alpha_m}
\end{align*}
\]

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 $i^*$):

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_{i^*-1}$</th>
<th>$A_{i^*}$</th>
<th>$A_{i^*+1}$</th>
<th>$A_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_p^{\alpha_1}$</td>
<td>$g_p^{\alpha_{i^*-1}}$</td>
<td>$g_p^{\alpha_{i^*}} g_q^{r}$</td>
<td>$g_p^{\alpha_{i^*+1}}$</td>
<td>$g_p^{\alpha_m}$</td>
</tr>
</tbody>
</table>

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

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

Extracted bit for a commitment $\sigma$ is 1 if $\sigma$ has a (non-zero) component in $\mathbb{G}_q$
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 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 \( \sigma_x = g_p^sg_q^t \)

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

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

\[ t^2 = tr \mod q \]
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 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 \( \mathbb{G} \).

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

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

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

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
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 component \( A \) part of the CRS (honestly-generated)

Similar analysis shows that extracted bits satisfy \( b_3 = 1 - b_1 b_2 = \text{NAND}(b_1, b_2) \)

[See paper for details]
A Commit-and-Prove Strategy for Batch Arguments

Let \( w_i = (w_{i,1}, \ldots, w_{i,m}) \) be a vector of wire labels associated with wire \( i \) across the \( m \) instances.

1. **Prover commits to each vector of wire assignments**

   \[
   w_i = w_{i,1} \quad w_{i,2} \quad \ldots \quad w_{i,m} \quad \sigma_i
   \]

   **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

   **Key idea:** Validity checks are quadratic and can be checked in the exponent.
Batch argument for NP from standard assumptions over bilinear maps

Subgroup decision assumption in composite-order bilinear groups

\[ G \cong G_p \times G_q \]

composite-order group

Simulate subgroups with subspaces

Conclusion:

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

Common reference string:

\[ A_1 A_2 \ldots A_m \]

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

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

\[ \vdots \]

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

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]
Choudhuri et al. [CJJ21] showed:

- Batch argument for NP*
- Somewhere extractable commitment

\[ \text{Delegation scheme for RAM programs} \]

\[ \text{succinct vector commitment that allows extracting on single index} \]

*Needs a split verification property [see paper for details]
Choudhuri et al. [CJJ21] showed:

- Batch argument for NP*

  *Needs a split verification property [see paper for details]

+ Somewhere extractable commitment

  succinct vector commitment that allows extracting on single index

→ Delegation scheme for RAM programs

succinct argument for polynomial-time computations
Choudhuri et al. [CJJ21] showed:

- Batch argument for NP*
  - *This work (from k-Lin)*

- Somewhere extractable commitment
  - *This work + [OPWW15] (from SXDH)*

\[\text{Delegation scheme for RAM programs}\]

*Needs a split verification property [see paper for details]*
Application to RAM Delegation (“SNARGs for P”)

Choudhuri et al. [CJJ21] showed:

Batch argument for NP* + 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]
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 in the plain model

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

Thank you!