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 \textbf{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: \[ |\pi| = \text{poly}(\lambda, \log m, |C|) \]

\( \lambda \) : security parameter

Proof size can scale with circuit size \( \text{not a SNARG for NP} \)
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
Goal: Amortize the Cost of NP Verification

Boolean circuit satisfiability

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

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

Boolean circuit satisfiability

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

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

Computational soundness: polynomial-time prover cannot convince verifier of \((x_1, ..., x_m)\) if there is any \(i \in [m]\) where \(x_i \not\in \mathcal{L}_C\)
Goal: Amortize the Cost of NP Verification

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

Focus: Non-interactive setting (proof is a single message)
Goal: Amortize the Cost of NP Verification

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

Prover and verifier have access to a common reference string (CRS)
An Application: Succinct Argument for P

Turing machine $M$, input $x$, time bound $T$

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

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

Verification time: running time of verifier is $\text{poly}(\lambda, |x|) + \text{poly}(\lambda, \log T)$
(Very) high-level idea:

Prover commits to the vector of computation states \((st_0, \ldots, st_T)\)

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

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

Prover constructs a batch argument that all \(T\) transitions are valid

Statements are indices \(1, \ldots, T\) and the NP relation is checking validity of step \(i\)
Batch Arguments for NP

Special case of succinct non-interactive arguments for NP (SNARGs)
Constructions rely on idealized models or knowledge assumptions or indistinguishability obfuscation

Batch arguments from correlation intractable hash functions
Sub-exponential DDH (in pairing-free groups) + QR (with $\sqrt{m}$ size proofs) [CJJ21a]
Learning with errors (LWE) [CJJ21b]

Batch arguments from pairing-based assumptions
Non-standard, but falsifiable $q$-type assumption on bilinear groups [KPY19]
This Work

New constructions of non-interactive batch arguments for NP

Batch arguments for NP from standard assumptions over bilinear maps

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

Key feature: Construction is “low-tech”

No heavy tools like correlation-intractable hash functions or probabilistically-checkable proofs
Direct “commit-and-prove” approach à la classic NIZK construction of Groth-Ostrovsky-Sahai

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

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

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

Previous bilinear map constructions: random oracle based [BGLS03]
A Commit-and-Prove Strategy for Batch Arguments

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

Prover commits to each vector of wire assignments

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

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

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

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

1. Prover commits to each vector of wire assignments

\[
  w_i = \begin{array}{c}
    w_{i,1} \\
    w_{i,2} \\
    \vdots \\
    w_{i,m}
  \end{array}
\]

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 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.

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

Requirements:

- **Input validity**
- **Wire validity**

Prover constructs the following proofs:

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

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

1. Prover commits to each vector of wire assignments

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

Let \( w_i \) be vector of wire labels associated with wire \( i \) across the \( m \) instances.

2. Prover constructs the following proofs:

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

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 a vector of wire labels associated with wire \( i \) across the \( m \) instances.

Prover constructs the following proofs:

1. Prover commits to each vector of wire assignments

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

   \[\sigma_i\]

   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.
A Commit-and-Prove Strategy for Batch Arguments

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

1. Prover commits to each vector of wire assignments $w_i$.

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

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

2. Prover constructs the following proofs:
   - Input validity
   - Wire validity
   - Gate validity
   - Output validity
Construction from Composite-Order Groups

Pedersen multi-commitments: \textit{(without} randomness\textit{)}

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 \( \mathbb{G}_p \)

\text{crs: sample } \alpha_1, \ldots, \alpha_m \leftarrow \mathbb{Z}_N \\
\text{output } A_1 \leftarrow g_p^{\alpha_1}, \ldots, A_m \leftarrow g_p^{\alpha_m}

\text{commitment to } x = (x_1, \ldots, x_m) \in \{0,1\}^m: \quad \sigma_x = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m} \\
\begin{bmatrix} \sigma_x \end{bmatrix} = \begin{bmatrix} \sum_{i \in [m]} \alpha_i x_i \end{bmatrix} \quad \text{(subset product of the } A_i\text{'s)}
Proving Relations on Committed Values

**Common reference string:**

\[
\begin{align*}
\left[ \alpha_1 \right] & \Rightarrow A_1 = g_p^{\alpha_1} \\
\vdots & \Rightarrow x \\
\left[ \alpha_m \right] & \Rightarrow A_m = g_p^{\alpha_m}
\end{align*}
\]

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

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

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

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

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

\[
e\left(\begin{bmatrix} x \end{bmatrix}, \begin{bmatrix} y \end{bmatrix}\right) \rightarrow \begin{bmatrix} xy \end{bmatrix}
\]

**Multiplies exponents in the target group**
**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 $(x_1, \ldots, x_m)$:**

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

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

**Wire validity**

Commitment for each wire is a commitment to a 0/1 vector $x \in \{0,1\}$ if and only if $x^2 = x$

**Approach:** consider the following pairing relations:

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

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

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

Common reference string:

\[
\begin{align*}
[\alpha_1] & \quad A_1 = g_1^{\alpha_1} \\
\vdots & \\
[\alpha_m] & \quad A_m = g_1^{\alpha_m}
\end{align*}
\]

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

\[
\begin{align*}
[\sum_{i \in [m]} \alpha_i x_i] & \\
\sigma_x & = A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m} \\
& = g_1^{\alpha_1 x_1 + \cdots + \alpha_m x_m}
\end{align*}
\]

Wire validity

Commitment for each wire is a commitment to a 0/1 vector \(x \in \{0,1\}\) if and only if \(x^2 = x\)

Approach: consider the following pairing relations:

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

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

non-cross terms

cross terms
Proving Relations on Committed Values

**Common reference string:**

- $[\alpha_1] = A_1 = g_p^{\alpha_1}$
- $[..]$
- $[\alpha_m] = A_m = g_p^{\alpha_m}$

**Wire validity**

Commitment for each wire is a commitment to a 0/1 vector $x \in \{0,1\}$ if and only if $x^2 = x$

**Approach:** consider the following pairing relations:

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

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

- $\Sigma_{i \neq j} \alpha_i \alpha_j x_i x_j$ non-cross terms
- $\Sigma_{i \neq j} \alpha_i \alpha_j x_i x_j$ 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:
\[
e(\sigma_x, \sigma_x) \text{ and } e(\sigma_x, \prod_{i \in [m]} A_i)
\]

\[
e\left( \left[ \sum_{i \in [m]} \alpha_i x_i \right], \left[ \sum_{i \in [m]} \alpha_i \right] \right) = \left[ \sum_{i \in [m]} \alpha_i^2 x_i \right] \times \left[ \sum_{i \neq j} \alpha_i \alpha_j x_i \right]
\]
non-cross terms

\[
e\left( \left[ \sum_{i \in [m]} \alpha_i x_i \right], \left[ \sum_{i \in [m]} \alpha_i x_i \right] \right) = \left[ \sum_{i \in [m]} \alpha_i^2 x_i^2 \right] \times \left[ \sum_{i \neq j} \alpha_i \alpha_j x_i x_j \right]
\]
non-cross terms

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

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

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

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

Augment CRS with cross-terms

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

Give prover ability to eliminate cross-terms only

\[\left[\sum_{i\neq j} \alpha_i \alpha_j (x_i - x_i x_j)\right] \quad \left[\alpha_i \alpha_j\right] = B_{i,j} = g_p^{\alpha_i \alpha_j} \quad \forall i \neq j\]
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 $g_p$

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

$$V = B_{i,j}^{x_i - x_i x_j}$$

$$e(g_p, V)$$

Overall verification relation:

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

$$A = \Pi_{i \in [m]} A_i$$
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 \( g_p \)

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

\[ V = B_{i,j}^{x_i - x_i x_j} \]

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 \( x_i^2 = x_i \)
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: $e(\sigma_x, \sigma_x) = e(\sigma_x, A)e(g_p, V)$

$A = \Pi_{i \in [m]} A_i$

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

Correction factor to correct for cross terms
Proving Relations on Committed Values

**Common reference string:**

\[
\begin{bmatrix}
\alpha_1 \\
\vdots \\
\alpha_m
\end{bmatrix}
\]

\[
A_1 = g_p^{\alpha_1}, \quad A_m = g_p^{\alpha_m}
\]

\[
\begin{bmatrix}
\alpha_1 + \cdots + \alpha_m
\end{bmatrix}
\]

\[
A = \prod_{i \in [m]} A_i
\]

\[
\begin{bmatrix}
\alpha_i \alpha_j
\end{bmatrix}
\]

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

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

\[
\begin{bmatrix}
\sum_{i \in [m]} \alpha_i x_i
\end{bmatrix}
\]

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

**Gate validity**

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

![NAND gate diagram]

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

Can leverage same approach as before:

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

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

\[
e(\sigma_{w_1}, \sigma_{w_2}) = e(g_p, g_p) \sum_{i \in [m]} \alpha_i^2 w_{1,i}w_{2,i} + \sum_{i \neq j} \alpha_i \alpha_j w_{1,i}w_{2,j}
\]
Proving Relations on Committed Values

**Common reference string:**

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

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!

**Gate validity**

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

\[
\text{NAND} \quad w_1 \quad w_2 \quad w_3
\]

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

Can leverage same approach as before:

\[
\begin{align*}
e(\sigma_{w_3}, A) &= e(g_p, g_p) \sum_{i \in [m]} \alpha_i^2 w_{3,i} + \sum_{i \neq j} \alpha_i \alpha_j w_{3,i} \\
e(A, A) &= e(g_p, g_p) \sum_{i \in [m]} \alpha_i^2 + \sum_{i \neq j} \alpha_i \alpha_j \\
e(\sigma_{w_1}, \sigma_{w_2}) &= e(g_p, g_p) \sum_{i \in [m]} \alpha_i^2 w_{1,i}w_{2,i} + \sum_{i \neq j} \alpha_i \alpha_j w_{1,i}w_{2,j}
\end{align*}
\]
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 $w_i$.

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

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

Overall proof size ($t$ wires, $s$ gates):

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

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

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

\[
\begin{align*}
[\sum_{i \in [m]} \alpha_i x_i] \\
\sigma_x &= A_1^{x_1} A_2^{x_2} \cdots A_m^{x_m} \\
&= g_{p}^{\alpha_1 x_1 + \cdots + \alpha_m x_m}
\end{align*}
\]

Soundness requires some care:

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

Soundness in GOS is possible by \textit{extracting} a witness from the commitment

For a false statement, no witness exists

\textbf{Our setting:} commitments are \textit{succinct} – cannot extract a full witness

\textbf{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

If proof \( \pi \) verifies, then we can extract a witness \( w_i \) such that \( C(x_i, w_i) = 1 \)
Local Extraction

Normal mode:

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

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

Extracting mode:

(extract on \(i^*\))

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

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 $i^*$):

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

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

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

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

\[ t^2 = tr \mod q \]
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 \( \sigma_x = g_p^s g_q^t \)

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

If wire validity checks pass, then \( t = b_ir \) 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

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

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

In the order-$q$ subgroup, exponents must satisfy:

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

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

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

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

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

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 arguments 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**

Yields a batch argument from

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

Common reference string:

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

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

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

\[
\vdots
\]

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

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]
The Base Case

\[ x_1, x_2, \ldots, x_\ell, x_{\ell+1}, x_{\ell+2}, \ldots, x_{2\ell}, \ldots, x_{\ell^2-\ell+1}, x_{\ell^2-\ell+2}, \ldots, x_{\ell^2} \]

Prove knowledge of a proof \( \pi_i \) for each batch of statements

\[ \ell = \sqrt{m} \]

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

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

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

Both batch arguments are on \( \ell = \sqrt{m} \) statements
Batch Arguments with Split Verification

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

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

\( \text{runs in time } \text{poly}(\lambda, m, n) \)

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

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

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

Preprocesses statements into a short verification key

Fast online verification

(Similar property from [CJJ21])
Recursive Bootstrapping

\[ x_1, x_2, \ldots, x_\ell \quad \rightarrow \quad \pi_1 \]
\[ x_{\ell+1}, x_{\ell+2}, \ldots, x_{2\ell} \quad \rightarrow \quad \pi_2 \]
\[ \vdots \quad \vdots \quad \vdots \]
\[ x_{\ell^2-\ell+1}, x_{\ell^2-\ell+2}, \ldots, x_{\ell^2} \quad \rightarrow \quad \pi_\ell \]

Prove knowledge of a proof \( \pi_i \) for each batch of statements

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

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

Batch argument used to check the relation
\[ \mathcal{R}((C, vk_1, \ldots, vk_\ell), (\pi_1, \ldots, \pi_\ell)) = 1 \]
if \( \text{OnlineVerify}(vk_i, C, \pi_i) = 1 \)

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

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

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

\( \ell = \sqrt{m} \)

Use batch argument on \( \ell = \sqrt{m} \) instances to prove each batch
Batch Arguments with Split Verification

Verifier checks the following:

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

\[ C \cdot \text{poly}(\lambda) \]

\[ nm \cdot \text{poly}(\lambda) \]

\[ |C| \cdot \text{poly}(\lambda) \]

constant number of group operations per wire/gate

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

(no more input validity checks)

Only depends on the statement!

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

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

\[ \text{GenVK} (\text{crs}, (x_1, ..., x_m)) \rightarrow (\sigma_1, ..., \sigma_n) \]
Corollary: Batch arguments for NP from standard assumptions over bilinear maps

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

For a proof on $m$ instances of length $n$:

- **CRS size:** $|\text{crs}| = m^{\varepsilon} \cdot \text{poly}(\lambda)$ for any constant $\varepsilon > 0$
- **Proof size:** $|\pi| = \text{poly}(\lambda, |C|)$
- **Verification time:** $|\text{Verify}| = \text{poly}(\lambda, n, m) + \text{poly}(\lambda, |C|)$
Choudhuri et al. [CJJ21] showed:

Batch argument with split verification

\[ \text{Somewhere extractable commitment} \]

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

\[ \text{succinct argument for polynomial-time computations} \]

\[ \text{succinct vector commitment that allows extracting on single index} \]
Application to RAM Delegation (“SNARGs for P”)

Choudhuri et al. [CJJ21] showed:

- Batch argument with split verification
- Somewhere extractable commitment

This work (from \(k\)-Lin)

\textit{succinct vector commitment that allows extracting on single index}

\textit{Delegation scheme for RAM programs}

\textit{succinct argument for polynomial-time computations}
Application to RAM Delegation (“SNARGs for P”)

Choudhuri et al. [CJJ21] showed:

- Batch argument with split verification
  
  *This work (from k-Lin)*

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

—

Delegation scheme for RAM programs
Choudhuri et al. [CJJ21] showed:

**Batch argument with split verification**

*This work (from $k$-Lin)*

**Somewhere extractable commitment**

*This work + [OPWW15] (from SXDH)*

**Delegation scheme for RAM programs**

**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 $(m_1, \ldots, m_k)$:

$$|\sigma^*| = \text{poly}(\lambda, \log k)$$
Application to Aggregate Signatures

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 batch argument for NP:

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

Can replace SNARKs for NP with a batch argument for NP:
prove knowledge of $\sigma_1, \ldots, \sigma_k$ such that $\text{Verify}(vk, m_i, \sigma_i) = 1$

**This work:** Batch argument 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]
Batch arguments for NP from standard assumptions over bilinear maps

**Key feature:** Construction is “low-tech”

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

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

**Corollary:** Aggregate signature with bounded aggregation

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

Thank you!