Succinct Functional Commitments for Circuits from $\kappa$-Lin

Hoeteck Wee and David Wu
June 2024
Functional Commitments

\[ \sigma \]

Commit

"commitment"

\[ f(x) \]

"opening"

Open + Verify

\[ \pi \]
Commitment (crs, x) → (σ, st)

Takes a common reference string and commits to an input x

Outputs commitment σ and commitment state st
Commit(crs, \(x\)) \rightarrow (\sigma, st)

Open(st, \(f\)) \rightarrow \pi

Takes the commitment state and a function \(f\) and outputs an opening \(\pi\)

Verify(crs, \(\sigma\), \((f, y)\), \(\pi\)) \rightarrow 0/1

Checks whether \(\pi\) is valid opening of \(\sigma\) to value \(y\) with respect to \(f\)
Functional Commitments

**Correctness:** if $(\sigma, \text{st}) \leftarrow \text{Commit}(\text{crs}, x)$ and $\pi \leftarrow \text{Open}(\text{st}, f)$
then $\text{Verify}(\text{crs}, \sigma, (f, f(x)), \pi) = 1$

*Can open commitment to $x$ to value $y = f(x)$ for any function $f$*
Functional Commitments

**Binding:** efficient adversary cannot open $\sigma$ to two different values with respect to the same $f$

- Open + Verify
  - $\sigma$ to $f(x)$

- Verify (crs, $\sigma$, $(f, y_0)$, $\pi_0$) = 1
- Verify (crs, $\sigma$, $(f, y_1)$, $\pi_1$) = 1
Succinctness: commitments and openings should be short

- **Short commitment**: $|\sigma| = \text{poly}(\lambda, \log |x|)$
- **Short opening**: $|\pi| = \text{poly}(\lambda, \log|x|)$
Special Cases of Functional Commitments

**Vector commitments:**

\[ [x_1, x_2, \ldots, x_n] \]

\[ \text{ind}_i(x_1, \ldots, x_n) := x_i \]

*commit to a vector, open at an index*

**Polynomial commitments:**

\[ [\alpha_0, \alpha_1, \ldots, \alpha_d] \]

\[ f_x(\alpha_0, \ldots, \alpha_d) := \alpha_0 + \alpha_1 x + \cdots + \alpha_d x^d \]

*commit to a polynomial, open to the evaluation at \( x \)*
Commitments as Proofs on Committed Data

Commitment as \( \text{Commit}(crs, \text{data}) \)

\[ \pi, f(\text{data}) \]

\( \pi \) is a proof that the data satisfies some property (e.g., committed input is in a certain range)

**Succinctness:** both the commitment and the proof are short.
# Succinct Functional Commitments

*(not an exhaustive list!)*

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Function Class</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Mer87]</td>
<td>vector commitment</td>
<td>collision-resistant hash functions</td>
</tr>
<tr>
<td>[LY10, CF13, LM19, GRWZ20]</td>
<td>vector commitment</td>
<td>$q$-type pairing assumptions</td>
</tr>
<tr>
<td>[CF13, LM19, BBF19]</td>
<td>vector commitment</td>
<td>groups of unknown order</td>
</tr>
<tr>
<td>[PPS21]</td>
<td>vector commitment</td>
<td>short integer solutions (SIS)</td>
</tr>
<tr>
<td>[KZG10, Lee20]</td>
<td>polynomial commitment</td>
<td>$q$-type pairing assumptions</td>
</tr>
<tr>
<td>[BFS19, BHRRS21, BF23]</td>
<td>polynomial commitment</td>
<td>groups of unknown order</td>
</tr>
<tr>
<td>[CLM23, FLV23]</td>
<td>polynomial commitment</td>
<td>$k$-R-ISIS assumption (lattices)</td>
</tr>
<tr>
<td>[LRY16]</td>
<td>linear functions</td>
<td>$q$-type pairing assumptions</td>
</tr>
<tr>
<td>[ACLMT22, CLM23]</td>
<td>constant-degree polynomials</td>
<td>$k$-R-ISIS assumption (lattices)</td>
</tr>
<tr>
<td>[LRY16]</td>
<td>Boolean circuits</td>
<td>collision-resistant hash functions + SNARKs</td>
</tr>
<tr>
<td>[dCP23]</td>
<td>Boolean circuits</td>
<td>SIS (non-succinct openings in general)</td>
</tr>
<tr>
<td>[KLVW23]</td>
<td>Boolean circuits</td>
<td>batch arguments for NP</td>
</tr>
<tr>
<td>[BCFL23]</td>
<td>Boolean circuits</td>
<td>twin $k$-R-ISIS (lattice) / HiKER (pairing)</td>
</tr>
<tr>
<td>[WW23a, WW23b]</td>
<td>Boolean circuits</td>
<td>$\ell$-succinct SIS</td>
</tr>
</tbody>
</table>
This work: functional commitments for general circuits using pairings

Why bilinear maps? Schemes have the best succinctness
- Pairing-based SNARKs just have a constant number of group elements

Can we construct a functional commitment for general circuits where the size of the commitment and the opening contain a constant number of group elements?

Namely: match the succinctness of pairing-based SNARKs, but only using standard pairing-based assumptions (no knowledge assumptions or ideal models)
Pairing-Based Functional Commitments

This work: functional commitments for **general circuits** using **pairings**

| Scheme       | Function Class          | $|\text{crs}|$   | $|\sigma|$ | $|\pi|$ | Assumption                                      |
|--------------|-------------------------|----------------|-------------|--------|------------------------------------------------|
| [LRY16, Gro16] | arithmetic circuits     | $O(s)$         | $O(1)$      | $O(1)$ | generic group                                   |
| [LRY16]      | linear functions        | $O(\ell)$      | $O(1)$      | $O(m)$ | subgroup decision                              |
| [LM19]       | linear functions        | $O(\ell m)$    | $O(1)$      | $O(1)$ | generic group                                   |
| [LP20]       | $\mu$-sparse polynomials | $O(\mu)$     | $O(m)$      | $O(1)$ | über assumption                                 |
| [CFT22]      | degree-$d$ polynomials  | $O(\ell^d m)$  | $O(d)$      | $O(d)$ | $\ell^d$-Diffie-Hellman exponent               |
| [BCFL23]     | arithmetic circuits     | $O(s^5)$       | $O(1)$      | $O(d)$ | hinted kernel ($q$-type)                        |
| [KLVW23]     | arithmetic circuits     | $\text{poly}(\lambda)$ | $O(1)$ | $\text{poly}(\lambda)$ | $k$-Lin                                      |

**This work**

| Function Class          | $|\text{crs}|$   | $|\sigma|$ | $|\pi|$ | Assumption                                      |
|-------------------------|----------------|-------------|--------|------------------------------------------------|
| arithmetic circuits     | $O(s^5)$       | $O(1)$      | $O(1)$ | bilateral $k$-Lin                               |

$\ell = \text{input length}, m = \text{output length}, s = \text{circuit size}$

metrics in # group elements
## This Work

**This work:** functional commitments for **general circuits** using **pairings**

| Scheme       | Function Class       | $|\text{crs}|$  | $|\sigma|$ | $|\pi|$ | Assumption        |
|--------------|----------------------|----------------|------------|--------|-------------------|
| This work    | arithmetic circuits  | $O(s^5)$       | $O(1)$     | $O(1)$ | bilateral $k$-Lin |

- First pairing-based construction for general **circuits** based on **falsifiable** assumptions where commitment and openings contain **constant** number of group elements
  - **Previously:** needed SNARKs (non-falsifiable assumptions)
- First scheme that only makes **black-box** use of cryptographic primitives/algorithms where the commitment + opening size is $\text{poly}(\lambda)$ bits
  - **Previously:** need non-black-box techniques (e.g., SNARKs or BARGs for NP)
This Work

**This work:** functional commitments for **general circuits** using **pairings**

| Scheme         | Function Class       | $|\text{crs}|$   | $|\sigma|$ | $|\pi|$ | Assumption       |
|----------------|----------------------|-----------------|-----------|--------|-----------------|
| This work      | arithmetic circuits  | $O(s^5)$        | $O(1)$    | $O(1)$ | bilateral $k$-Lin |

Additional implications (for free!):
- SNARG for P/poly with a **universal** setup with constant-size proofs (CRS only depends on the size of the circuit)
  - **Previously (from pairings):** SNARG for P/poly with circuit-dependent CRS [GZ21]
- Homomorphic signature for general (bounded-size) circuits with constant-size signatures
  - **Previously (from pairings):** Signature size scaled with the *depth* of the circuit [BCFL23]

*(all results without relying on knowledge assumptions or ideal models)*
Chainable commitment [BCFL23]

Let $f : \mathbb{Z}_p^\ell \to \mathbb{Z}_p^d$ be a vector-valued function.

Can think of commitment as a subset product:

$$\sigma = \prod_{i \in [\ell]} h_i^{x_i}$$

where $h_i$ are in the CRS.

succinct commitment to vector $x$

Open to commitment to $y = f(x)$

Chain binding: cannot open $\sigma_{\text{in}}$ to two distinct commitments $\sigma_{\text{out}}, \sigma'_{\text{out}}$

succinct commitment to vector $y = f(x)$
Chainable commitment for **quadratic functions** $\Rightarrow$ functional commitment for **circuits**

*Assume:* each gate computes quadratic function

Commit to input wires $\sigma$

Chainable commitment openings for each layer

Commitments to internal layers and output layer

[BCFL23]
Chainable commitment for **quadratic functions** $\Rightarrow$ functional commitment for **circuits**

Commitment: $\sigma$
Opening: $(\sigma_1', \sigma_2', \sigma_3', \pi_1, \pi_2, \pi_3)$

Opening scales with depth of circuit

Commit to input wires $\sigma$

Commitments to internal layers and output layer

Chainable commitment openings for each layer

[BCFL23]
Our Approach: Commit to All Wires

Goal: Constant number of group elements for commitment and openings

Commitment: (same as before)

Verifier know output \((z_1, \ldots, z_d)\):

Opening: commit to all wires (i.e., concatenated together) twice

\[
\begin{align*}
x_1 & \quad x_2 & \quad \ldots & \quad x_\ell & \quad y_1 & \quad y_2 & \quad \ldots & \quad y_t & \quad z_1 & \quad z_2 & \quad \ldots & \quad z_d & \quad \sigma_1 \\
x_1 & \quad x_2 & \quad \ldots & \quad x_\ell & \quad y_1 & \quad y_2 & \quad \ldots & \quad y_t & \quad z_1 & \quad z_2 & \quad \ldots & \quad z_d & \quad \sigma_2
\end{align*}
\]
**Our Approach: Commit to All Wires**

**Goal:** Constant number of group elements for commitment and openings

<table>
<thead>
<tr>
<th>Input layer</th>
<th>Intermediate layer</th>
<th>Output layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$, $x_2$, ..., $x_{\ell}$</td>
<td>$y_1$, $y_2$, ..., $y_t$</td>
<td>$z_1$, $z_2$, ..., $z_d$</td>
</tr>
</tbody>
</table>

**Commitment:** (same as before)

- $x_1$, $x_2$, ..., $x_{\ell}$

**Verifier know output ($z_1$, ..., $z_d$):**

- $z_1$, $z_2$, ..., $z_d$  

**Opening:** commit to all wires (i.e., concatenated together) twice

- $x_1$, $x_2$, ..., $x_{\ell}$, $z_1$, $z_2$, ..., $z_d$  

Everything is short, but how do we argue binding?
Our Approach: Commit to All Wires

**Goal:** Constant number of group elements for commitment and openings

**Commitment:** (same as before)

\[ \sigma_{\text{in}} \]

\[ x_1 \ x_2 \ \cdots \ x_\ell \]

Verifier know output \((z_1, \ldots, z_t): \)

\[ \sigma_{\text{out}} \]

\[ z_1 \ z_2 \ \cdots \ z_d \]

**Opening:** commit to all wires (i.e., concatenated together) **twice**

\[ x_1 \ x_2 \ \cdots \ x_\ell \ y_1 \ y_2 \ \cdots \ y_t \ z_1 \ z_2 \ \cdots \ z_d \]

\[ \sigma_1 \]

Neither \(\sigma_1\) nor \(\sigma_2\) is a quadratic function of \(\sigma_{\text{input}}\)

With bilinear maps, we only know how to check quadratic functions

\[ \sigma_2 \]
Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) and \((\sigma'_1, \sigma'_2, \sigma'_{\text{out}}, \pi')\)

Initially: no guarantees on what \(\sigma_1, \sigma'_1, \sigma_2, \sigma'_2\) commit to

Cannot use chain binding to argue that \(\sigma_1\) and \(\sigma'_1\) are equal since they are not a quadratic function of \(\sigma_{\text{in}}\)

Our approach: argue that a prefix of \(\sigma_1, \sigma'_1\) are still equal
Consider two different openings: \((σ_1, σ_2, σ_{out}, π)\) and \((σ'_1, σ'_2, σ'_{out}, π')\).

\[
\begin{array}{cccccc}
\sigma_1 & σ_1' & σ_2 & σ_2' & \ldots & \sigma_\ell
\end{array}
\]

Initially: no guarantees on what \(σ_1, σ_1', σ_2, σ_2'\) commit to

Input consistency: \(π, π'\) includes an opening that asserts that the first \(ℓ\) components of \(σ_1, σ_1'\) are consistent with \(σ_{in}\)
Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) and \((\sigma'_1, \sigma'_2, \sigma'_{\text{out}}, \pi')\)

Close to a chain binding property: prover is opening \(\sigma_{\text{in}}\) to output commitments \(\sigma_1, \sigma'_1\)

**Caveat:** Only reasoning about the first \(\ell\) components of \(\sigma_1\) and \(\sigma'_1\) (*not* the entire vector)

**Input consistency:** \(\pi, \pi'\) includes an opening that asserts that the first \(\ell\) components of \(\sigma_1, \sigma'_1\) are consistent with \(\sigma_{\text{in}}\)
Consider two different openings: $(\sigma_1, \sigma_2, \sigma_{out}, \pi)$ and $(\sigma'_1, \sigma'_2, \sigma'_{out}, \pi')$

If we establish that the first $\ell$ components of $\sigma_1, \sigma'_1$ agree, we can try to argue that the first $\ell + 1$ components of $\sigma_2, \sigma'_2$ also agree.

Observation: first $\ell + 1$ components of $\sigma_2, \sigma'_2$ is a quadratic function of the first $\ell$ components of $\sigma_1, \sigma'_1$
Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) and \((\sigma'_1, \sigma'_2, \sigma'_{\text{out}}, \pi')\)

If we establish that the first \(\ell\) components of \(\sigma_1, \sigma'_1\) agree, we can try to argue that the first \(\ell + 1\) components of \(\sigma_2, \sigma'_2\) also agree.

**Observation:** first \(\ell + 1\) components of \(\sigma_2, \sigma'_2\) is a quadratic function of the first \(\ell\) components of \(\sigma_1, \sigma'_1\)
Consider two different openings: $(\sigma_1, \sigma_2, \sigma_{out}, \pi)$ and $(\sigma'_1, \sigma'_2, \sigma'_{out}, \pi')$

Repeat this process: if $\sigma_2, \sigma'_2$ agree on the first $\ell + 1$ values, then $\sigma_1, \sigma'_1$ agree on the first $\ell + 1$ values
Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) and \((\sigma_1', \sigma_2', \sigma_{\text{out}}', \pi')\)

Repeat this process: if \(\sigma_2, \sigma_2'\) agree on the first \(\ell + 1\) values, then \(\sigma_1, \sigma_1'\) agree on the first \(\ell + 1\) values.
Approach Overview

Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) and \((\sigma'_1, \sigma'_2, \sigma'_{\text{out}}, \pi')\)

Iterate to conclude that \(\sigma_1, \sigma'_1\) actually agree on all values (including the outputs), which implies binding
Prove statements of the following form:

- **Input consistency**: first $\ell$ wires in $\sigma_1$ is consistent with $\sigma_{in}$
- **Gate consistency**: first $j + 1$ wires in $\sigma_2$ is consistent with first $j$ wires in $\sigma_1$
- **Internal consistency**: first $j$ wires in $\sigma_1$ is consistent with first $j$ wires in $\sigma_2$
- **Output consistency**: last $t$ wires in $\sigma_1$ are consistent with $\sigma_{out}$

Main technical tool: way to reason about prefixes of a committed vector
Projective Chainable Commitments

Intuitively: can associate CRS with an index $j$ that allows projecting a commitment $\sigma_1$ onto a commitment to the first $j$ indices.

Projective chain binding: given $(\sigma_1, \sigma_2, \pi)$ and $(\sigma_1', \sigma_2', \pi')$ if $\text{Project}(td, \sigma_1, j) = \text{Project}(td, \sigma_1', j)$ and

- $(\sigma_2, \pi, f)$ is a valid opening for $\sigma_1$
- $(\sigma_2', \pi', f)$ is a valid opening for $\sigma_1'$

Then, $\text{Project}(td, \sigma_2, j + 1) = \text{Project}(td, \sigma_2', j + 1)$
Using Projective Chainable Commitments

Consider two different openings: \((σ_1, σ_2, σ_{out}, π)\) and \((σ'_1, σ'_2, σ'_{out}, π')\)

Initially: no guarantees on what \(σ_1, σ'_1, σ_2, σ'_2\) commit to

Step 1: Input consistency between \(σ_{in}\) and \(σ_1, σ'_1\)

Projective chain binding: \(σ_1, σ'_1\) are both openings for \(σ_{in}\) so \(\text{Project}(σ_1, ℓ) = \text{Project}(σ'_1, ℓ)\)
Using Projective Chainable Commitments

Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) and \((\sigma'_1, \sigma'_2, \sigma'_{\text{out}}, \pi')\)

<table>
<thead>
<tr>
<th>(\hat{x}_1)</th>
<th>(\hat{x}_2)</th>
<th>(\ldots)</th>
<th>(\hat{x}_\ell)</th>
<th>(\sigma_1, \sigma'_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma_1)</td>
<td>(\sigma'_1)</td>
<td>(\sigma_2)</td>
<td>(\sigma'_2)</td>
<td>(\sigma_{\text{out}})</td>
</tr>
</tbody>
</table>

\(\sigma_1\) and \(\sigma'_1\) agree on first \(\ell\) components: 
Project(\(\sigma_1, \ell\)) = Project(\(\sigma'_1, \ell\))

Note: we do not know what values they have, only that they agree

<table>
<thead>
<tr>
<th>(\sigma_2)</th>
<th>(\sigma'_2)</th>
<th>(\sigma_{\text{out}})</th>
</tr>
</thead>
</table>

Step 1: Input consistency between \(\sigma_{\text{in}}\) and \(\sigma_1, \sigma'_1\)

Projective chain binding: \(\sigma_1, \sigma'_1\) are both openings for \(\sigma_{\text{in}}\) so Project(\(\sigma_1, \ell\)) = Project(\(\sigma'_1, \ell\))
Using Projective Chainable Commitments

Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) and \((\sigma'_1, \sigma'_2, \sigma'_{\text{out}}, \pi')\)

\[
\begin{array}{cccc}
\hat{x}_1 & \hat{x}_2 & \cdots & \hat{x}_\ell \\
\end{array}
\]

\[
\begin{array}{cccc}
\sigma_1 \quad \sigma'_1 \\
\end{array}
\]

\[
\begin{array}{cccc}
\sigma_2 \quad \sigma'_2 \\
\end{array}
\]

\[
\begin{array}{cccc}
\sigma_{\text{in}} \\
\end{array}
\]

\(\sigma_1\) and \(\sigma'_1\) agree on first \(\ell\) components: \(\text{Project}(\sigma_1, \ell) = \text{Project}(\sigma'_1, \ell)\)

\[\text{Note: we do not know what values they have, only that they agree}\]

Step 2: Gate consistency between first \(\ell\) wires in \(\sigma_1, \sigma'_1\) with first \(\ell + 1\) wires in \(\sigma_2, \sigma'_2\)

Since \(\text{Project}(\sigma_1, \ell) = \text{Project}(\sigma'_1, \ell)\), projective chain binding implies \(\text{Project}(\sigma_2, \ell + 1) = \text{Project}(\sigma'_2, \ell + 1)\)
Using Projective Chainable Commitments

Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{out}, \pi)\) and \((\sigma_1', \sigma_2', \sigma_{out}', \pi')\)

\[
\begin{array}{cccc}
\hat{x}_1 & \hat{x}_2 & \cdots & \hat{x}_\ell \\
\end{array}
\]

\[
\begin{array}{cccc}
\tilde{x}_1 & \tilde{x}_2 & \cdots & \tilde{x}_\ell \\
\end{array}
\]

\[
\begin{array}{cccc}
\check{x}_1 & \check{x}_2 & \cdots & \check{x}_\ell \\
\end{array}
\]

\[
\begin{array}{cccc}
\bar{x}_1 & \bar{x}_2 & \cdots & \bar{x}_\ell \\
\end{array}
\]

\[
\begin{array}{cccc}
\tilde{y}_1 & \\
\end{array}
\]

Step 2: Gate consistency between first \(k\) wires in \(\sigma_1, \sigma_1'\) with first \(\ell + 1\) wires in \(\sigma_2, \sigma_2'\)

Since \(\text{Project}(\sigma_1, \ell) = \text{Project}(\sigma_1', \ell)\), projective chain binding implies \(\text{Project}(\sigma_2, \ell + 1) = \text{Project}(\sigma_2', \ell + 1)\)
Using Projective Chainable Commitments

Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{out}, \pi)\) and \((\sigma'_1, \sigma'_2, \sigma'_{out}, \pi')\).

\[
\begin{array}{cccc}
\hat{x}_1 & \hat{x}_2 & \ldots & \hat{x}_\ell \\
\sigma_1, \sigma'_1 \\
\sigma_2, \sigma'_2
\end{array}
\]

\(\sigma_2\) and \(\sigma'_2\) agree on first \(\ell + 1\) components:

\[
\text{Project}(\sigma_2, \ell + 1) = \text{Project}(\sigma'_2, \ell + 1)
\]

**Step 3:** Internal consistency between first \(\ell + 1\) wires in \(\sigma_2, \sigma'_2\) with first \(\ell + 1\) wires in \(\sigma_1, \sigma'_1\).

Since \(\text{Project}(\sigma_2, \ell + 1) = \text{Project}(\sigma'_2, \ell + 1)\), projective chain binding implies \(\text{Project}(\sigma_1, \ell + 1) = \text{Project}(\sigma'_1, \ell + 1)\).
Using Projective Chainable Commitments

Consider two different openings: $(\sigma_1, \sigma_2, \sigma_{out}, \pi)$ and $(\sigma'_1, \sigma'_2, \sigma'_{out}, \pi')$

\[\begin{array}{cccc}
\hat{x}_1 & \hat{x}_2 & \cdots & \hat{x}_\ell & \hat{y}_1 \\
\end{array}\]

$\sigma_1$ and $\sigma'_1$ agree on first $\ell + 1$ components:
Project($\sigma_1, \ell + 1$) = Project($\sigma'_1, \ell + 1$)

\[\begin{array}{cccc}
\tilde{x}_1 & \tilde{x}_2 & \cdots & \tilde{x}_\ell & \tilde{y}_1 \\
\end{array}\]

\[\begin{array}{cccc}
\hat{x}_1 & \hat{x}_2 & \cdots & \hat{x}_\ell & \hat{y}_1 \\
\end{array}\]

Step 3: Internal consistency between first $\ell + 1$ wires in $\sigma_2, \sigma'_2$ with first $\ell + 1$ wires in $\sigma_1, \sigma'_1$

Since Project($\sigma_2, \ell + 1$) = Project($\sigma'_2, \ell + 1$), projective chain binding implies Project($\sigma_1, \ell + 1$) = Project($\sigma'_1, \ell + 1$)
Using Projective Chainable Commitments

Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) and \((\sigma'_1, \sigma'_2, \sigma'_{\text{out}}, \pi')\)

\[\sigma_1, \sigma'_1\]

\[\sigma_2, \sigma'_2\]

Observe: we have established that Project\((\sigma_1, \ell + 1) = \text{Project}(\sigma'_1, \ell + 1)\)

Can iterate this strategy for each index \(\ell + 1, \ell + 2, \ldots\) to argue that \(\sigma_1, \sigma'_1\) agree on all components.
Using Projective Chainable Commitments

Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{out}, \pi)\) and \((\sigma'_1, \sigma'_2, \sigma'_{out}, \pi')\)

\[
\begin{array}{cccc}
\hat{x}_1 & \hat{x}_2 & \cdots & \hat{x}_\ell \\
\hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_t \\
\hat{z}_1 & \hat{z}_2 & \cdots & \hat{z}_d \\
\end{array}
\]

\[
\begin{array}{cccc}
\bar{x}_1 & \bar{x}_2 & \cdots & \bar{x}_\ell \\
\bar{y}_1 & \bar{y}_2 & \cdots & \bar{y}_t \\
\bar{z}_1 & \bar{z}_2 & \cdots & \bar{z}_d \\
\end{array}
\]

\[
\sigma_1, \sigma'_1 \\
\sigma_2, \sigma'_2
\]

**Observe:** we have established that \(\text{Project}(\sigma_1, \ell + 1) = \text{Project}(\sigma'_1, \ell + 1)\)
Can iterate this strategy for each index \(\ell + 1, \ell + 2, \ldots\) to argue that \(\sigma_1, \sigma'_1\) agree on all components
Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) and \((\sigma_1', \sigma_2', \sigma_{\text{out}}', \pi')\)

If \(\sigma_1 = \sigma_1'\), then final output commitment check ensures \(\sigma_{\text{out}} = \sigma_{\text{out}}'\)

Similar proof strategy as [GZ21, CJJ21, KLVW23]
Constructing Projective Chainable Commitments

Starting point: Kiltz-Wee [KW15] proof system for proving membership in linear spaces

Suppose we want to support openings to a fixed linear function

\[ x \in \mathbb{Z}_p^\ell \rightarrow Mx \in \mathbb{Z}_p^d \] where \( M \in \mathbb{Z}_p^{d \times \ell} \)

Let \((\mathbb{G}, \mathbb{G}_T, e)\) be a pairing group and let \( g \) be a generator of \( \mathbb{G} \)

Common reference string contains two vectors \( g^t \) and \( g^{\hat{t}} \) where \( t \leftarrow \mathbb{Z}_p^\ell \) and \( \hat{t} \leftarrow \mathbb{Z}_p^d \)

Vector \( t \) is used to commit to the inputs and vector \( \hat{t} \) is used to commit to outputs

Commitment to input \( x \in \mathbb{Z}_p^\ell \) is \( \sigma_{\text{in}} = g^{t^T x} \)

Commitment to output \( y \in \mathbb{Z}_p^d \) is \( \sigma_{\text{out}} = g^{\hat{t}^T y} \)

Basically a Pedersen (vector) commitment:
if \( g^t = [h_1, \ldots, h_\ell] \), then \( \sigma = \prod_{i \in [\ell]} h_i^{x_i} \)
Suppose we want to support openings to a fixed linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d \times \ell} \]

Commitment to input \( x \in \mathbb{Z}_p^\ell \) is \( \sigma_{\text{in}} = g^{t^T x} \) Commitment to output \( y \in \mathbb{Z}_p^d \) is \( \sigma_{\text{out}} = g^{\widehat{t^T y}} \)

To support openings to the linear function \( M \) \((x \mapsto Mx)\), we also include in the CRS \( g^z^T \) where

\[ z^T = wt^T - r\widehat{t}^T M \in \mathbb{Z}_p^\ell \text{ and } r, w \leftarrow \mathbb{Z}_p \]
Suppose we want to support openings to a fixed linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d \times \ell} \]

Commitment to input \( x \in \mathbb{Z}_p^\ell \) is \( \sigma_{in} = g^{t^T x} \)

Commitment to output \( y \in \mathbb{Z}_p^d \) is \( \sigma_{out} = g^{\hat{t}^T y} \)

To support openings \( (x \mapsto Mx) \), we also include in the CRS \( g^{z^T} \) where

\[ z^T = wt^T - r\hat{t}^T M \in \mathbb{Z}_p^\ell \quad \text{and} \quad r, w \leftarrow \mathbb{Z}_p \]

Intuitively: \( z \) “recodes” an input commitment with respect to \( t \) to an output commitment with respect to \( \hat{t} \).
Suppose we want to support openings to a \textit{fixed} linear function

\[
x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d
\]

where \( M \in \mathbb{Z}_p^{d \times \ell} \)

Commitment to input \( x \in \mathbb{Z}_p^\ell \) is \( \sigma_{\text{in}} = g^{t^Tx} \)  
Commitment to output \( y \in \mathbb{Z}_p^d \) is \( \sigma_{\text{out}} = g^{\hat{t}^Ty} \)

To support openings to the linear function \( M \ (x \mapsto Mx) \), we also include in the CRS \( g^{z^T} \) where

\[
z^T = wt^T - r\hat{t}^TM \in \mathbb{Z}_p^\ell \quad \text{and} \quad r, w \leftarrow \mathbb{Z}_p
\]

For now, we consider the \textbf{designated-verifier} setting where \textbf{secret key} needed to check proofs

\begin{align*}
\text{Opening: } \pi &= g^{z^Tx} \\
\text{Verification relation: } &\text{Check that } \pi = \frac{\sigma_{\text{in}}^w}{\sigma_{\text{out}}^r} \\
\text{Secret verification key: } &r, w \\
\text{Correctness: } &\frac{\sigma_{\text{in}}^w}{\sigma_{\text{out}}^r} = \frac{g^{wt^Tx}}{g^{\hat{t}^Ty}} = \frac{g^{wt^Tx}}{g^{\hat{t}^TMx}} = g^{(wt^T-r\hat{t}^TM)x} = g^{z^Tx} = \pi
\end{align*}
Security for Linear Functions

Suppose we want to support openings to a *fixed* linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d\times\ell} \]

**Common reference string:** \( g^t, g^\hat{t}, g^{wt^T - r\hat{t}^T}M \)

**Verification relation:** Check that \( \pi = \frac{\sigma_w}{\sigma_{\text{out}}} \)

Suppose adversary produces the following:

- Input commitment \( \sigma_{\text{in}} = g^c \)
- Output commitments \( \sigma_{\text{out}} = g^{\hat{c}}, \sigma'_{\text{out}} = g^{\hat{c}'} \)
- Openings \( \pi = g^v, \pi' = g^{v'} \)

If the openings are valid, then

\[ v = wc - r\hat{c} \]
\[ v' = wc - r\hat{c}' \]

Thus, \( v - v' = r(\hat{c} - \hat{c}') \)

Non-zero since \( \hat{c} \neq \hat{c}' \)
Security for Linear Functions

Suppose we want to support openings to a *fixed* linear function

\[ \mathbf{x} \in \mathbb{Z}_{p^\ell} \mapsto \mathbf{Mx} \in \mathbb{Z}_{p^d} \]

**Common reference string:** \( g^t, \hat{g}^t, g^{wt-T}r^TM \)

**Verification relation:** Check that \( \pi = \frac{\sigma_{in}^w}{\sigma_{out}^r} \)

Suppose adversary produces the following:

- Input commitment \( \sigma_{in} = g^c \)
- Output commitments \( \sigma_{out} = g^\hat{c}, \sigma'_{out} = g^\hat{c'} \)
- Openings \( \pi = g^v, \pi' = g^{v'} \)

Under DDH, \( wt \) computationally *hides* value of \( r \)

**Technically:** DDH does not hold in a symmetric pairing group, but can use asymmetric group (or k-Lin)

Distribution of \( r(\hat{c} - \hat{c}') \) is pseudorandom from the perspective of the adversary, so this check passes with probability \( 1/p \)

Thus, \( v - v' = r(\hat{c} - \hat{c}') \)

Non-zero since \( \hat{c} \neq \hat{c}' \)
Chainable Commitments for Linear Functions

Suppose we want to support openings to a \textit{fixed} linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d \times \ell} \]

Common reference string: \( g^t, \hat{g}^t, g^{wt^T - r\hat{t}^T} M \)

Verification relation: Check that \( \pi = \frac{\sigma_{in}}{\sigma_{out}} \)

\[ \sigma_{in} = g^{t^Tx} \]
\[ \sigma_{out} = g^{\hat{t}^Ty} \]

Lots of caveats:
- Only supports \textit{fixed} functions
- Only supports \textit{linear} functions
- Only \textit{designated-verifier}
Chainable Commitments for Linear Functions

Suppose we want to support openings to a \textit{fixed} linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d \times \ell} \]

\textbf{Common reference string:} \( g^t, g^\hat{t}, g^{wt^T - r\hat{t}^TM} \)
\[ \sigma_{\text{in}} = g^{t^Tx} \]
\[ \sigma_{\text{out}} = g^{\hat{t}^Ty} \]

\textbf{Verification relation:} Check that \( \pi = \frac{\sigma_{\text{in}}}{\sigma_{\text{out}}} \)

\[ \sigma_{\text{in}} = g^{t^Tx} \]
\[ \sigma_{\text{out}} = g^{\hat{t}^Ty} \]

\textbf{Caveat:} Only supports \textit{fixed} functions

Extend to arbitrary functions by relying on \textbf{linear homomorphism}

Suppose we publish \( g^{z_1^T} = g^{w_1t^T - r\hat{t}^TM_1} \) and \( g^{z_2^T} = g^{w_2t^T - r\hat{t}^TM_2} \) in the CRS

\[ \sigma_{\text{in}} = g^{t^Tx} \]
\[ g^{\alpha_1z_1^T}x \text{ is an opening to } y = \alpha_1 M_1 x \]
\[ \sigma_{\text{out}} = g^{\hat{t}^Ty} \]
\[ \frac{\sigma_{\text{in}}}{\sigma_{\text{out}}} = g^{\alpha_1w_1 t^Tx - r\hat{t}^Ty} = g^{\alpha_1w_1 t^Tx - \alpha_1 \hat{t}^TM_1 x} = g^{\alpha_1 z_1^Tx} \]
Chainable Commitments for Linear Functions

Suppose we want to support openings to a fixed linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d \times \ell} \]

Common reference string: \( g^t, g^{\hat{t}}, g^{wtT-rtTM} \)

Verification relation: Check that \( \pi = \frac{\sigma_{\text{in}}^w}{\sigma_{\text{out}}^r} \)

\( \sigma_{\text{in}} = g^{t^Tx} \quad \sigma_{\text{out}} = g^{\hat{t}^Ty} \)

**Caveat:** Only supports fixed functions

Extend to arbitrary functions by relying on linear homomorphism

Suppose we publish \( g^{z_1^T} = g^{w_1tT-rtTM_1} \) and \( g^{z_2^T} = g^{w_2tT-rtTM_2} \) in the CRS

\( \sigma_{\text{in}} = g^{t^Tx} \quad g^{\alpha_1z_1^Tx} \) is an opening to \( \alpha_1M_1x \)

\( \sigma_{\text{out}} = g^{\hat{t}^Ty} \quad g^{\alpha_2z_2^Tx} \) is an opening to \( \alpha_2M_2x \)
**Chainable Commitments for Linear Functions**

\[ \frac{\sigma_{\text{in}}^{\alpha_1 w_1}}{g r \hat{r}^T (\alpha_1 M_1 x)} = g^{\alpha_1 z_1^T} x \]

\[ \frac{\sigma_{\text{in}}^{\alpha_2 w_2}}{g r \hat{r}^T (\alpha_2 M_2 x)} = g^{\alpha_2 z_2^T} x \]

**Caveat:** Only supports **fixed** functions

Extend to arbitrary functions by relying on linear homomorphism

Suppose we publish \( g z_1^T = g^{w_1 T} t^T - r T M_1 \) and \( g z_2^T = g^{w_2 T} t^T - r T M_2 \) in the CRS

\( \sigma_{\text{in}} = g t^T x \)

\( g^{\alpha_1 z_1^T x + \alpha_2 z_2^T x} \) is an opening to \( y = \alpha_1 M_1 x + \alpha_2 M_2 x \)

\( \sigma_{\text{out}} = g \hat{r}^T y \)

\( \frac{\sigma_{\text{in}}^{\alpha_1 w_1 + \alpha_2 w_2}}{\sigma_{\text{out}}^{r}} = g^{\alpha_1 z_1^T x + \alpha_2 z_2^T x} \)

**Verification relation for** \( x \mapsto (\alpha_1 M_1 + \alpha_2 M_2) x \)**
Chainable Commitments for Linear Functions

\[
\sigma_{\text{in}}^{\alpha_1 w_1} = g^{\alpha_1 z_1^T x} \\
g^{r \hat{r}^T (\alpha_1 M_1 x)}
\]

\[
\sigma_{\text{in}}^{\alpha_2 w_2} = g^{\alpha_2 z_2^T x} \\
g^{r \hat{r}^T (\alpha_2 M_2 x)}
\]

**Caveat:** Only supports **fixed** functions

Extend to arbitrary functions by relying on **linear homomorphism**

Publish components for complete basis of linear functions

\[
M_{i,j} = \begin{bmatrix}
0 & \cdots & 0 \\
\vdots & 1 & \vdots \\
0 & \cdots & 0
\end{bmatrix}_{\text{row } i, \text{column } j}
\]

Any linear function \( M \) can be expressed as a linear combination of \( M_{i,j} \)
Chainable Commitments for Linear Functions

Suppose we want to support openings to a fixed linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d \times \ell} \]

**Common reference string:** \( g^t, g^\hat{t}, g^{wt^T-r\hat{t}M} \)

**Verification relation:** Check that \( \pi = \frac{\sigma_{in}}{\sigma_{out}} \)

\[ \sigma_{in} = g^{t^Tx} \]
\[ \sigma_{out} = g^{\hat{t}^Ty} \]

**Caveat:** Only supports linear functions

Can extend to quadratic functions by linearization (and tensoring)

Quadratic function of \( x \) is a linear function of \( x \otimes x \)

[see paper for details]
Chainable Commitments for Linear Functions

Suppose we want to support openings to a *fixed* linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \] where \( M \in \mathbb{Z}_p^{d \times \ell} \)

**Common reference string:** \( g^t, g^{\hat{t}}, g^{wt^T-r\hat{t}^TM} \)

**Verification relation:** Check that \( \pi = \frac{\sigma_{in}}{\sigma_{out}} \)

\[ \sigma_{in} = g^{t^Tx} \]
\[ \sigma_{out} = g^{\hat{t}^Ty} \]

**Caveat:** Only **designated-verifier**

**Solution:** encode the verification key \( r \) and \( w \) in the exponent (following [KW15])
Chainable Commitments for Linear Functions

Suppose we want to support openings to a *fixed* linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d \times \ell} \]

**Common reference string:** \( \mathcal{g}^g_T, \mathcal{g}^w_t, \mathcal{g}^{\mathcal{w}^T} \)

**Verification relation:** Check that \( \pi = \sigma \) in \( \sigma \text{ out} \)

**Caveat:** Only designated-verifier

**Solution:** encode the verification key \( r \) and \( w \) in the exponent (following [KW15])

Previous argument required that \( r \) was computationally hidden, so we cannot just give out \( g^r \)
Chainable Commitments for Linear Functions

Suppose we want to support openings to a fixed linear function

\[ x \in \mathbb{Z}_p^\ell \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d \times \ell} \]

Common reference string: \( g^t, g^\hat{t}, g^{wt^T - r\hat{t}^TM} \) \[ \sigma_{in} = g^{t^Tx} \]

Verification relation: Check that \( \pi = \frac{\sigma_{in}^w}{\sigma_{out}^r} \) \[ \sigma_{out} = g^{\hat{t}^Ty} \]

Caveat: Only designated-verifier

Solution: encode the verification key \( r \) and \( w \) in the exponent (following [KW15])

Sample \( a \leftarrow \mathbb{Z}_p^2 \) \[ \text{CRS: } g^t, g^\hat{t}, g^aw, g^{a^Tw}, g^{a^Tr}, g^{wt^T - r\hat{t}^TM} \]

Sample \( w, r \leftarrow \mathbb{Z}_p^2 \) \[ \text{Verification relation is now} \]

\[ \sigma_{in} = g^{t^Tx} \quad \sigma_{out} = g^{\hat{t}^TMx} \quad e\left(g^{a^T}, \pi\right) = \frac{e\left(g^{a^Tw}, \sigma_{in}\right)}{e\left(g^{a^Tr}, \sigma_{out}\right)} \quad \pi = g^{wt^T x - r\hat{t}^TMx} \]
Suppose we want to support openings to a fixed linear function

\[ x \in \mathbb{Z}_p^l \mapsto Mx \in \mathbb{Z}_p^d \text{ where } M \in \mathbb{Z}_p^{d \times l} \]

Common reference string: \( g^t, \hat{g}^\hat{t}, g^{wt^T-r\hat{t}^TM} \)  
\( \sigma_{in} = g^{t^Tx} \)

Verification relation: Check that \( \pi = \frac{\sigma_{in}^w}{\sigma_{out}^w} \)

Caveat: Only designated-verifier

Solution: encode the verification key \( r \) and \( w \) in the exponent (following \([KW15]\))

Sample \( a \leftarrow \mathbb{Z}_p^2 \)  
Sample \( w, r \leftarrow \mathbb{Z}_p^2 \)

CRS: \( g^t, \hat{g}^\hat{t}, g^a, g^{a^Tw}, g^{a^Tr}, g^{wt^T-r\hat{t}^TM} \)

Verification relation is now

\[ \sigma_{in} = g^{t^Tx} \quad \sigma_{out} = g^{\hat{t}^TMx} \quad e\left(g^{a^T}, \pi\right) = \frac{e\left(g^{a^Tw}, \sigma_{in}\right)}{e\left(g^{a^Tr}, \sigma_{out}\right)} \quad \pi = g^{wt^Tx-r\hat{t}^TMx} \]
Need a way to project a commitment onto a subset of its components

\[ g^t = [h_1, \ldots, h_\ell] \]

\[ \sigma = g^{t^T x} = \prod_{i \in [\ell]} h_i^{x_i} \]

In **composite-order groups**: introduce a subgroup for components in projection set

Suppose \( \mathbb{G} \) has order \( N = pq \) and let \( \mathbb{G}_p, \mathbb{G}_q \) be the order-\( p \) and order-\( q \) subgroups of \( \mathbb{G} \)

Let \( g_p \) be a generator of \( \mathbb{G}_p \) and \( g_q \) be a generator of \( \mathbb{G}_q \)

Replace \( g^t \) with \( h_1 = (g_p g_q)^{t_1}, \ldots, h_j = (g_p g_q)^{t_j}, h_{j+1} = g_p^{t_{j+1}}, \ldots, h_\ell = g_p^{t_\ell} \)
Projective Chainable Commitments

Need a way to project a commitment onto a subset of its components

\[ g^t = [h_1, \ldots, h_\ell] \]

\[ \sigma = g^{t^T x} = \prod_{i \in [\ell]} h_i^{x_i} \]

In composite-order groups: introduce a subgroup for components in projection set

Suppose \( G \) has order \( N = pq \) and let \( G_p, G_q \) be the order-\( p \) and order-\( q \) subgroups of \( G \)

Let \( g_p \) be a generator of \( G_p \) and \( g_q \) be a generator of \( G_q \)

Replace \( g^t \) with \( h_1 = (g_p g_q)^{t_1}, \ldots, h_j = (g_p g_q)^{t_j}, h_{j+1} = g_p^{t_{j+1}}, \ldots, h_{\ell} = g_p^{t_\ell} \)

Commitment is now

\[ \sigma = \prod_{i \in [\ell]} h_i^{x_i} = \prod_{i=1}^{j} (g_p g_q)^{t_i x_i} \prod_{i=j+1}^{\ell} g_p^{t_i x_i} \]

If we consider \( \sigma \) in the mod-\( q \) subgroup, then

\[ \sigma_q = \prod_{i \in [j]} g_q^{t_i x_i} \]

This is precisely a commitment to the first \( j \) components!
Projective Chainable Commitments

Commitment is now

$$\sigma = \prod_{i \in [\ell]} h_i^{x_i} = \prod_{i=1}^{j} (g_p g_q)^{t_i x_i} \prod_{i=j+1}^{\ell} g_p^{t_i x_i}$$

If we consider $\sigma$ in the mod-$q$ subgroup, then

$$\sigma_q = \prod_{i \in [j]} g_q^{t_i x_i}$$

This is precisely a commitment to the first $j$ components!

**Syntactic issue:** We were considering linear/quadratic functions over $\mathbb{Z}_p$ before; when using composite-order groups, we should view it as functions over the integers

**Main idea:** embed two copies of the chainable commitment scheme:

- The normal scheme is embedded in the $\mathbb{G}_p$-subgroup
- The projected scheme is embedded in the $\mathbb{G}_q$-subgroup

When reasoning about chain binding, we implement the previous proof argument within the $\mathbb{G}_q$ subgroup
Projective Chainable Commitments

Commitment is now
\[
\sigma = \prod_{i \in [\ell]} h_i^{x_i} = \prod_{i=1}^{j} (g_p g_q)^{t_i x_i} \prod_{i=j+1}^{\ell} g_p^{t_i x_i}
\]
If we consider \(\sigma\) in the mod-\(q\) subgroup, then
\[
\sigma_q = \prod_{i \in [j]} g_q^{t_i x_i}
\]
This is precisely a commitment to the first \(j\) components!

**Syntactic issue:** We were considering linear/quadratic functions over \(\mathbb{Z}_p\) before; when using composite-order groups, we should view it as functions over the integers

**Main idea:** embed two copies of the chainable commitment scheme:
- The normal scheme is embedded in the \(\mathbb{G}_p\)-subgroup
- The projected scheme is embedded in the \(\mathbb{G}_q\)-subgroup

**In paper:** use prime-order groups and consider two orthogonal subspaces (normal scheme in one subspace and projected scheme in the other); security reduces to (bilateral) \(k\)-Lin

[see paper for details; see also [GZ21] for similar projection approach]
**Goal:** Constant number of group elements for commitment **and** openings

**Commitment:**

\[ \sigma_{in} \]

Verifier know output \((z_1, ..., z_d)\):

\[ \sigma_{out} \]

**Opening:** commit to all wires (i.e., concatenated together) **twice**

Use projective chain binding and an iterative argument to argue binding
Summary

**This work:** functional commitments for **general circuits** using **pairings**

| Scheme         | Function Class  | $|\text{crs}|$  | $|\sigma|$ | $|\pi|$ | Assumption            |
|----------------|-----------------|-----------|----------|-------|-----------------------|
| This work      | arithmetic circuits | $O(s^5)$  | $O(1)$  | $O(1)$ | bilateral $k$-Lin     |

- First pairing-based construction for general **circuits** based on **falsifiable** assumptions where commitment and openings contain **constant** number of group elements
- First scheme that only makes **black-box** use of cryptographic primitives/algorithms where the commitment + opening size is $\text{poly}(\lambda)$ bits

**Open problem:** Construction with shorter CRS (e.g., linear-size)? Then, parameters would match state-of-the-art pairing-based SNARKs

**Thank you!**

https://eprint.iacr.org/2024/688