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

Hoeteck Wee and David Wu
May 2024
Functional Commitments

$\mathbf{x}$

Commit

$\sigma$

“commitment”

Open + Verify

$\sigma$

“opening”

$\pi$

$f(x)$
Functional Commitments

Commit \((\text{crs}, x) \rightarrow (\sigma, \text{st})\)

Takes a common reference string and commits to an input \(x\)

Outputs commitment \(\sigma\) and commitment state \(\text{st}\)
**Functional Commitments**

\[ \sigma \xrightarrow{\text{Open} + \text{Verify}} f(x) \]

\[ \text{Commit}(\text{crs}, x) \rightarrow (\sigma, \text{st}) \]

\[ \text{Open}(\text{st}, f) \rightarrow \pi \]

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

\[ \text{Verify}(\text{crs}, \sigma, (f, y), \pi) \rightarrow 0/1 \]

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

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

\[
\begin{align*}
\pi_0 & \quad (f, y_0) & \quad \text{Verify}(\text{crs}, \sigma, (f, y_0), \pi_0) = 1 \\
\pi_1 & \quad (f, y_1) & \quad \text{Verify}(\text{crs}, \sigma, (f, y_1), \pi_1) = 1
\end{align*}
\]
Functional Commitments

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:

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

ind\(_i\)(x\(_1\), \ldots, x\(_n\)) := x\(_i\)

commit to a vector, open at an index

Polynomial commitments:

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

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

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

\[ \text{Commit}(\text{crs}, \text{data}) \]

\[ \sigma \]

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

\( \pi \) 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>
Pairing-Based Functional Commitments

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 assumption (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)            |
| [KLW23]          | arithmetic circuits  | poly($\lambda$) | $O(1)$      | poly($\lambda$) | $k$-Lin                           |
| **This work**    | 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: 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 \cite{BCFL23}

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

Can think of commitment as a subset product:

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

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 $\sigma_{\text{out}}, \sigma'_{\text{out}}$
Chainable commitment for **quadratic functions** $\Rightarrow$ functional commitment for **circuits**

**Assume:** each gate computes quadratic function.

Commit to input wires $\sigma$.

Commitments to internal layers and output layer.

Chainable commitment openings for each layer.

References: [BCFL23]
Starting Point: Chainable Commitment

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)

\[ x_1, x_2, \ldots, x_k \rightarrow \sigma_{\text{input}} \]

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

\[ z_1, z_2, \ldots, z_t \rightarrow \sigma_{\text{output}} \]

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

\[ x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_\ell, z_1, z_2, \ldots, z_t \rightarrow \sigma_1 \]

\[ \rightarrow \sigma_2 \]
**Our Approach: Commit to All Wires**

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

Commitment: (same as before)

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

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

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)

\[x_1 x_2 \ldots x_k \rightarrow \sigma_{\text{input}}\]

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

\[z_1 z_2 \ldots z_t \rightarrow \sigma_{\text{output}}\]

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

\[x_1 x_2 \ldots x_k y_1 y_2 \ldots y_\ell z_1 z_2 \ldots z_t \rightarrow \sigma_1\]

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

\[x_1 x_2 \ldots x_k y_1 y_2 \ldots y_\ell z_1 z_2 \ldots z_t \rightarrow \sigma_2\]
Technical Tool: 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.

Vanilla chain binding: given $(\sigma_1, \sigma_2, \pi)$ and $(\sigma_1', \sigma_2', \pi')$

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

Then, $\sigma_2 = \sigma_2'$
Technical Tool: 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

Prove statements of the following form:

- **Input consistency**: first $k$ wires in $\sigma_1$ is consistent with $\sigma_{\text{input}}$
- **Gate consistency**: first $j + 1$ wires in $\sigma_2$ is consistent with first $j$ wires in $\sigma_1$
Prove statements of the following form:

- **Input consistency**: first $k$ wires in $\sigma_1$ is consistent with $\sigma_{\text{input}}$
- **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 $\sigma_{\text{output}}$

This is a quadratic relation (since we have the intermediate wires)
Using Projective Chainable Commitments

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

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

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

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

\[
\begin{array}{cccc}
\hat{x}_1 & \hat{x}_2 & \ldots & \hat{x}_k \\
\end{array}
\]

\[
\begin{array}{cccc}
\sigma_1, \sigma_1' \\
\sigma_2, \sigma_2' \\
\end{array}
\]

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

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

**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 \((\sigma_1, k) = \text{Project}(\sigma_1', k)\)
Using Projective Chainable Commitments

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

\(\hat{x}_1 \hat{x}_2 \ldots \hat{x}_k \rightarrow \sigma_{\text{in}}\)

\(\sigma_1 \) and \(\sigma'_1 \) agree on first \(k\) components: 

\[ \text{Project}(\sigma_1, k) = \text{Project}(\sigma'_1, k) \]

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

\(\sigma_2, \sigma'_2\)

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

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

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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\cdots$</th>
<th>$x_k$</th>
<th>$\sigma_{\text{in}}$</th>
</tr>
</thead>
</table>

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

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

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

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

\(\sigma_2\) and \(\sigma_2'\) agree on first \(k + 1\) components:
\[
\text{Project}(\sigma_2, k + 1) = \text{Project}(\sigma_2', k + 1)
\]

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

\(\sigma_2, \sigma_2'\)

**Step 3:** Internal consistency between first \(k + 1\) wires in \(\sigma_2, \sigma_2'\) with first \(k + 1\) wires in \(\sigma_1, \sigma_1'\)

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

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

\[
\begin{array}{cccc}
\hat{x}_1 & \hat{x}_2 & \ldots & \hat{x}_k & \hat{y}_1 \\
\end{array}
\]

\(\sigma_1, \sigma'_1\)

\(\sigma_1\) and \(\sigma'_1\) agree on first \(k + 1\) components:

Project\((\sigma_1, k + 1) = \text{Project}(\sigma'_1, k + 1)\)

\[
\begin{array}{cccc}
\hat{x}_1 & \hat{x}_2 & \ldots & \hat{x}_k & \hat{y}_1 \\
\end{array}
\]

\(\sigma_2, \sigma'_2\)

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

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

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

\(\hat{x}_1 \hat{x}_2 \ldots \hat{x}_k \hat{y}_1 \quad \sigma_1, \sigma'_1\)

\(\hat{x}_1 \hat{x}_2 \ldots \hat{x}_k \hat{y}_1 \quad \sigma_1, \sigma'_1\)

\(\hat{x}_1 \hat{x}_2 \ldots \hat{x}_k \hat{y}_1 \quad \sigma_2, \sigma'_2\)

\(\hat{x}_1 \hat{x}_2 \ldots \hat{x}_k \hat{y}_1 \quad \sigma_2, \sigma'_2\)

\(\sigma_1 \text{ and } \sigma'_1 \text{ agree on first } k + 1 \text{ components:}\)

\(\text{Project}(\sigma_1, k + 1) = \text{Project}(\sigma'_1, k + 1)\)

\(\sigma_1 \text{ and } \sigma'_1 \text{ agree on first } k + 1 \text{ components:}\)

\(\text{Project}(\sigma_1, k + 1) = \text{Project}(\sigma'_1, k + 1)\)

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

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

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

Observe: we have established that \(\text{Project}(\sigma_1, k + 1) = \text{Project}(\sigma'_1, k + 1)\)
Can iterate this strategy for each index \(k + 1, k + 2, \ldots\) to argue that \(\sigma_1, \sigma'_1\) agree on all components
Using Projective Chainable Commitments

Consider two different openings: \((\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)\) 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
  - Basic scheme supports opening to a **fixed** linear function
  - Extend to **any** linear function using multiple copies of the scheme (for basis functions)
  - Extend to quadratic functions via tensoring and linearization

**Projective chainable commitments:** embed commitment in a vector space
  - Real commitment lie in one subspace, projected commitment lie in a “shadow” subspace
    *similar projection as [GZ19], but with additional locality constraints*

Security follows from bilateral $k$-Lin

[see paper for details]
This work: functional commitments for **general circuits** using **pairings**

| Scheme          | Function Class    | |crs|   | |σ|   | |π|   | 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