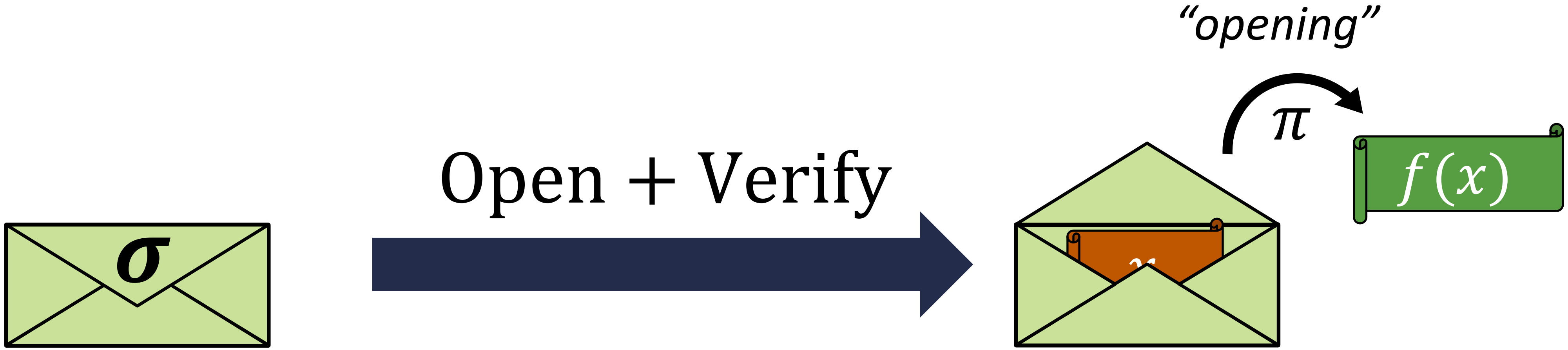
Succinct Functional Commitments for Circuits from k-Lin

Hoeteck Wee and David Wu

May 2024

Functional Commitments



Functional Commitments

$$\mathrm{Commit}(\mathrm{crs}, x) \to (\sigma, \mathrm{st})$$

Takes a common reference string and commits to an input x

Outputs commitment σ and commitment state st

Functional Commitments

$\mathsf{Commit}(\mathsf{crs}, x) \to (\sigma, \mathsf{st})$

$\mathsf{Open}(\mathsf{st}, f) \to \pi$

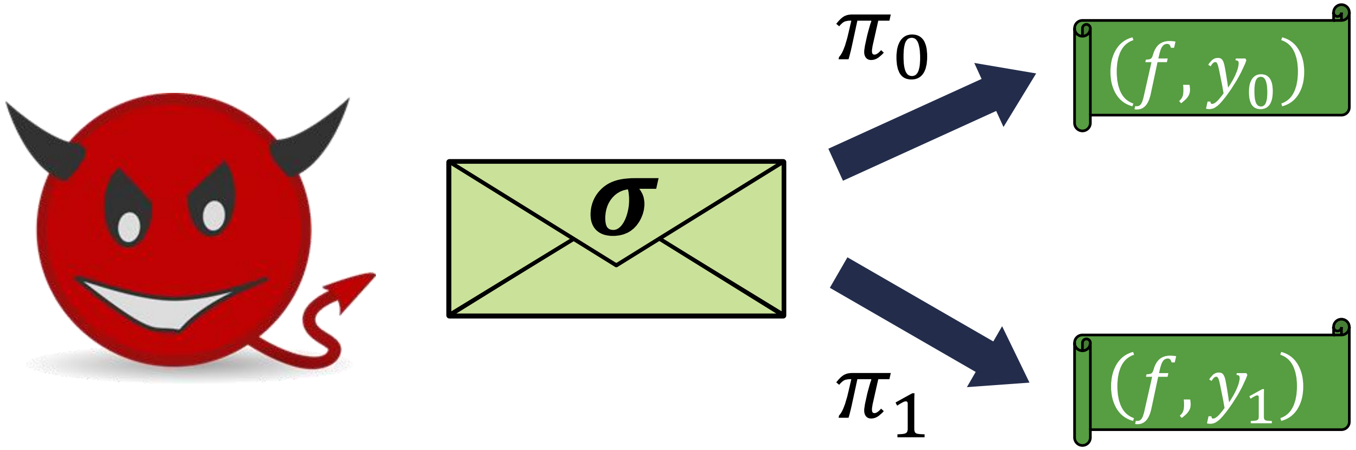
Takes the commitment state and a function f and outputs an opening π

$\mathsf{Verify}(\mathsf{crs}, \sigma, (f, y), \pi) \to 0/1$

Checks whether π is valid opening of σ to value y with respect to f

Functional Commitments

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



$$\text{Verify}(\text{crs}, \sigma, (f, y_0), \pi_0) = 1$$

$$\text{Verify}(\text{crs}, \sigma, (f, y_1), \pi_1) = 1$$

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:

$[x_1, x_2, \dots, x_n]$ $\xrightarrow{\mathrm{ind}_i(x_1, \dots, x_n) \coloneqq x_i}$ x_i

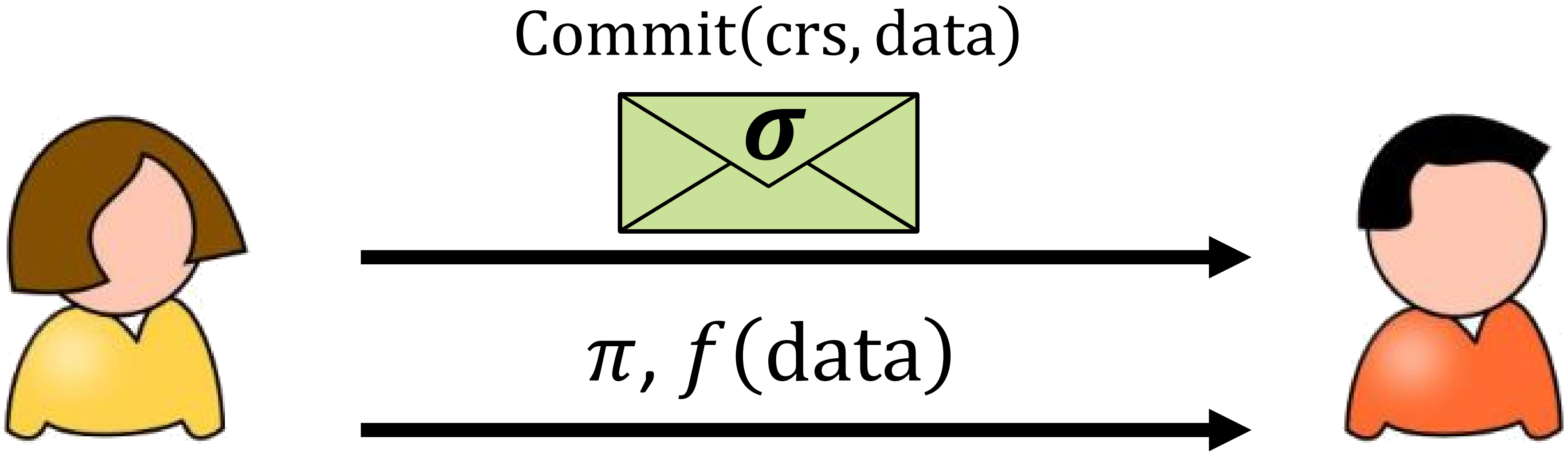
commit to a vector, open at an index

Polynomial commitments:

$[\alpha_0, \alpha_1, \dots, \alpha_d]$ $\xrightarrow{f_x(\alpha_0, \dots, \alpha_d) \coloneqq \alpha_0 + \alpha_1 x + \cdots + \alpha_d x^d}$ y

commit to a polynomial, open to the evaluation at x

Commitments as Proofs on Committed Data



π 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!)

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

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**	$\lvert\text{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\pi\rvert$	**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]	μ-sparse polynomials	$O(\mu)$	$O(m)$	$O(1)$	über assumption
[CFT22]	degree-d polynomials	$O(\ell^d m)$	$O(d)$	$O(d)$	ℓ^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	**arithmetic circuits**	$\boldsymbol{O(s^5)}$	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$	**bilateral k-Lin**

ℓ = input length, m = output length, s = circuit size

metrics in # group elements

This Work

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

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

Constant number of group elements

Additional implications (for free!):

- SNARG for $\mathrm{P/poly}$ with a **universal** setup with constant-size proofs (CRS only depends on the size of the circuit)
  - **Previously (from pairings):** SNARG for $\mathrm{P/poly}$ 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)

Starting Point: Chainable Commitment

Chainable commitment [BCFL23]

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

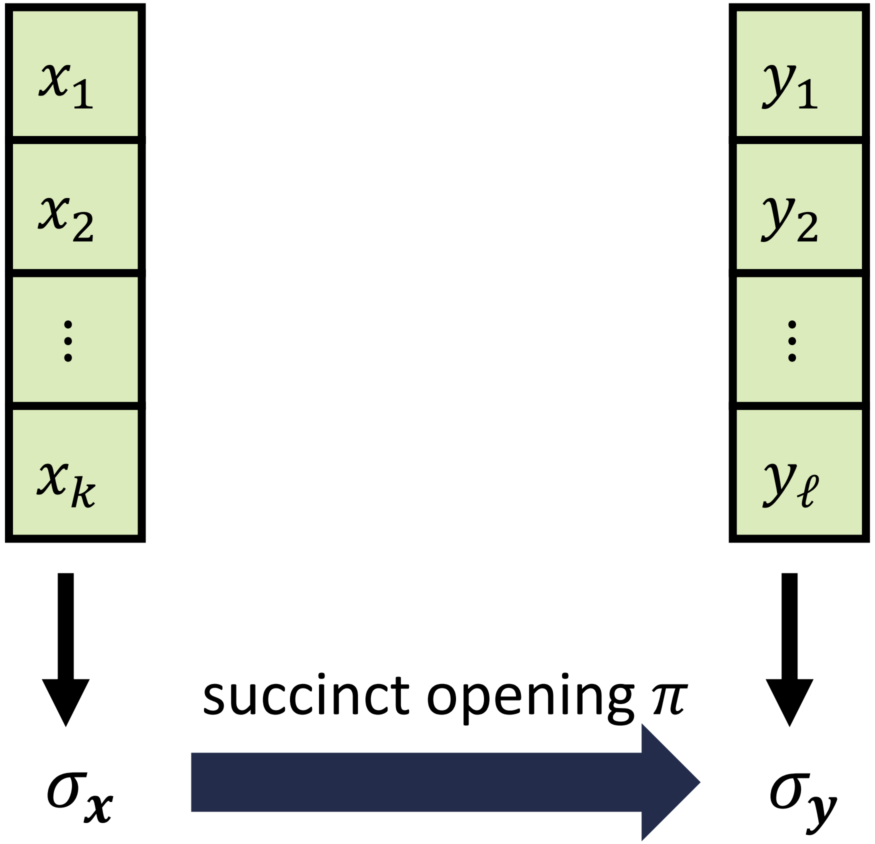
Instead of committing to $\boldsymbol{x}$ and opening to $\boldsymbol{y} = f(\boldsymbol{x})$

Open to **commitment** to $\boldsymbol{y} = f(\boldsymbol{x})$

Can think of commitment as a subset product:

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

where h_i are in the CRS



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

succinct commitment to vector $\boldsymbol{x}$

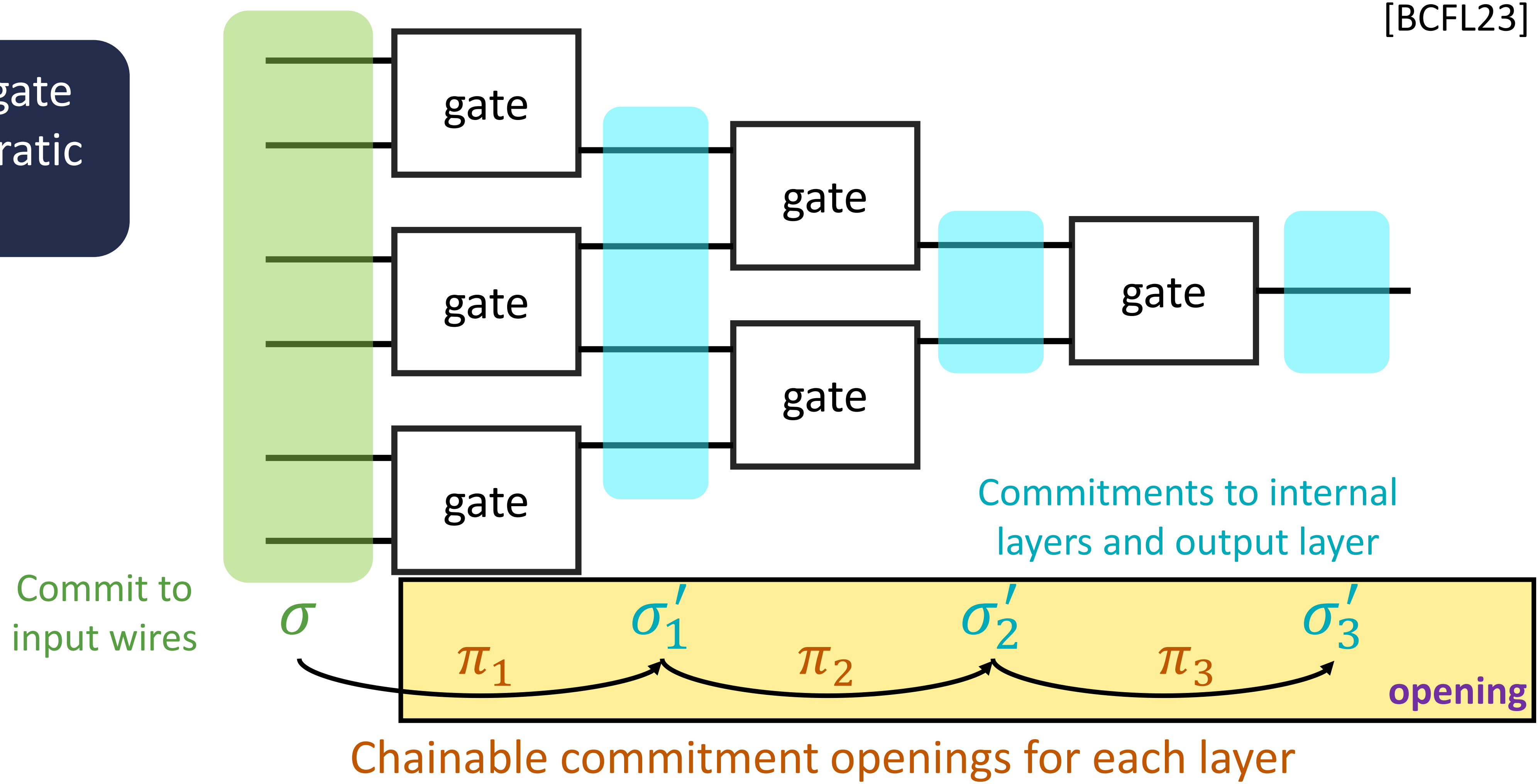
succinct commitment to vector $\boldsymbol{y} = f(\boldsymbol{x})$

Starting Point: Chainable Commitment

Chainable commitment for **quadratic functions** ⇒ functional commitment for **circuits**

[BCFL23]

Assume: each gate computes quadratic function



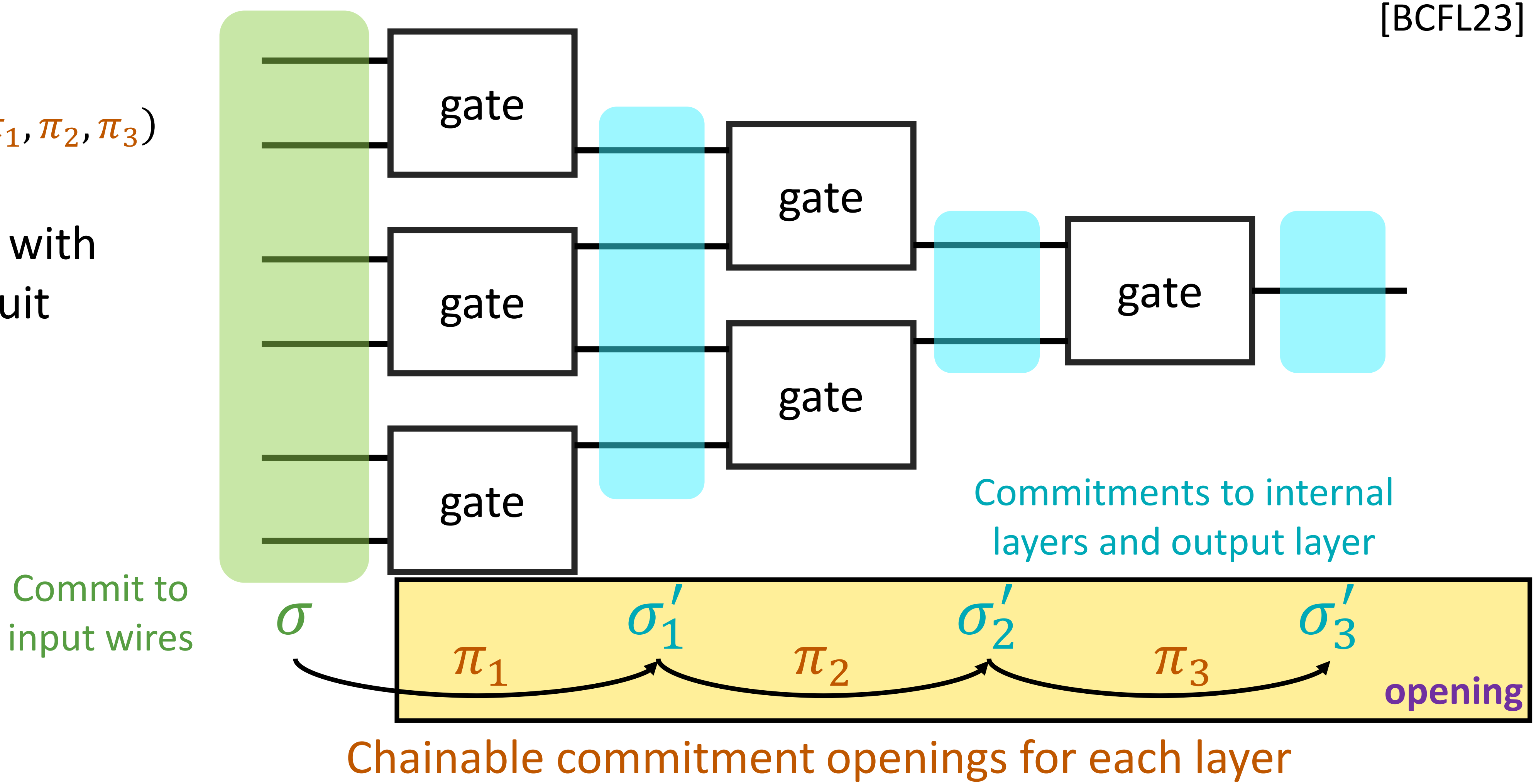
Starting Point: Chainable Commitment

Chainable commitment for **quadratic functions** ⇒ functional commitment for **circuits**

[BCFL23]

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

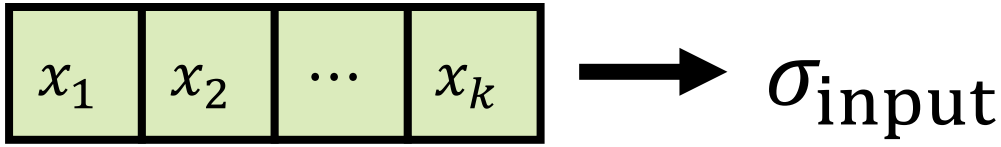
Opening scales with depth of circuit



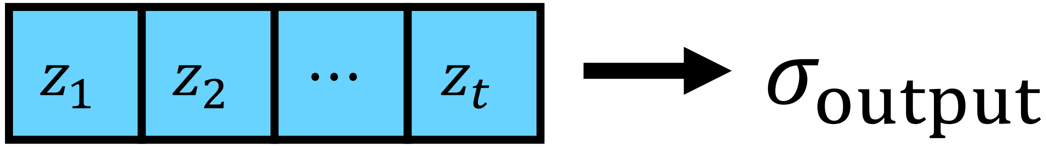
Our Approach: Commit to All Wires

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

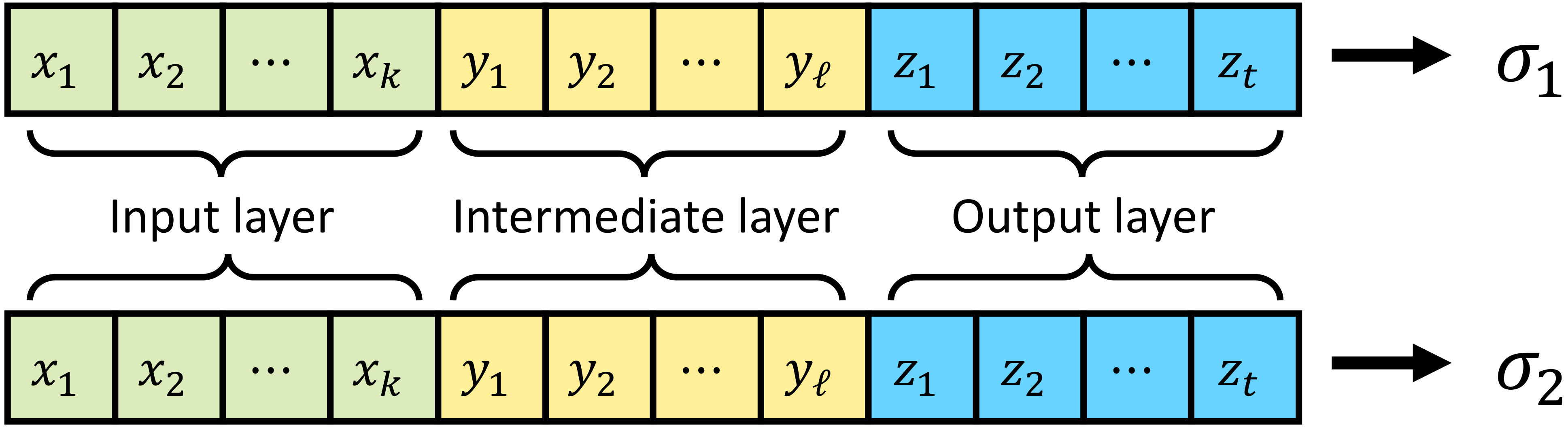
Commitment: (same as before)



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



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



Our Approach: Commit to All Wires

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

Commitment: (same as before)

x_1 | x_2 | $\cdots$ | x_k → σ_{input}

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

z_1 | z_2 | $\cdots$ | z_t → σ_{output}

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

x_1 | x_2 | $\cdots$ | x_k | y_1 | y_2 | $\cdots$ | y_ℓ | z_1 | z_2 | $\cdots$ | z_t → σ_1

Input layer — Intermediate layer — Output layer

x_1 | x_2 | $\cdots$ | x_k | y_1 | y_2 | $\cdots$ | y_ℓ | z_1 | z_2 | $\cdots$ | z_t → σ_2

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)

$[x_1 \mid x_2 \mid \cdots \mid x_k] \rightarrow \sigma_{\text{input}}$

Verifier know output $(z_1, \dots, z_t)$:

$[z_1 \mid z_2 \mid \cdots \mid z_t] \rightarrow \sigma_{\text{output}}$

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

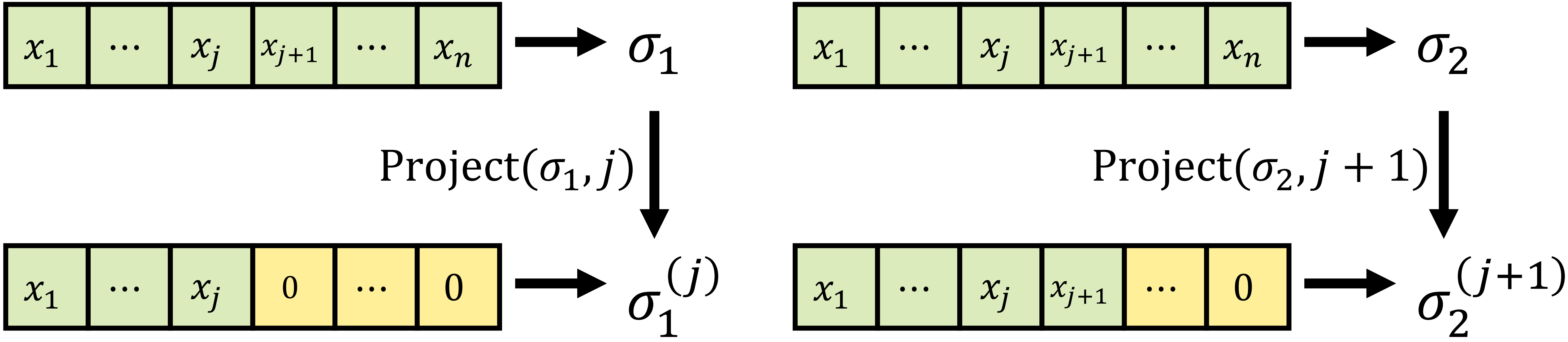
$[x_1 \mid x_2 \mid \cdots \mid x_k \mid y_1 \mid y_2 \mid \cdots \mid y_\ell \mid z_1 \mid z_2 \mid \cdots \mid z_t] \rightarrow \sigma_1$

Neither σ_1 nor σ_2 is a quadratic function of σ_{input}

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

$[x_1 \mid x_2 \mid \cdots \mid x_k \mid y_1 \mid y_2 \mid \cdots \mid y_\ell \mid z_1 \mid z_2 \mid \cdots \mid z_t] \rightarrow \sigma_2$

Technical Tool: Projective Chainable Commitments



Intuitively: can associate CRS with an index j that allows projecting a commitment σ_1 onto a commitment to the first j indices

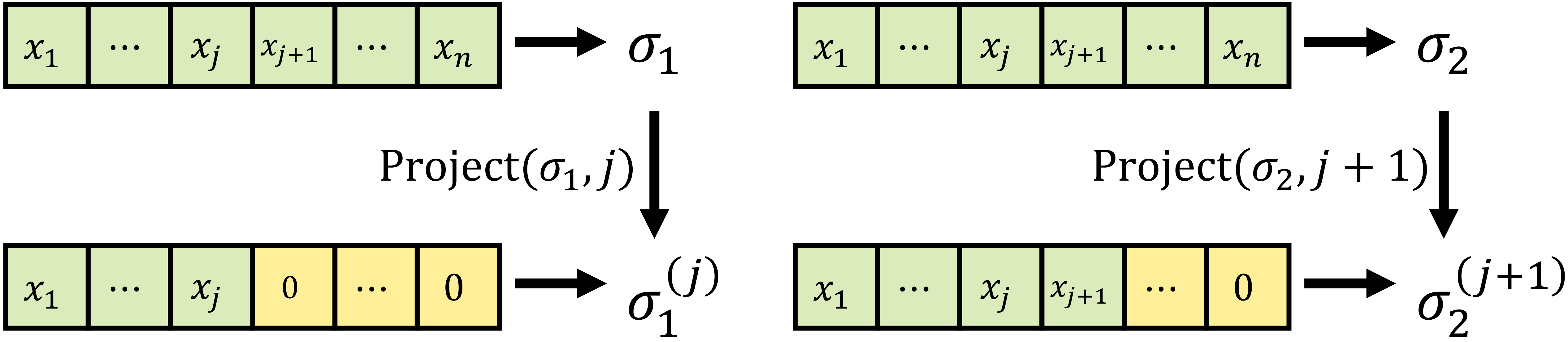
Vanilla chain binding: given $(\sigma_1, \sigma_2, \pi)$ and $(\sigma_1', \sigma_2', \pi')$

If $\sigma_1 = \sigma_1'$ and

- (σ_2, π, f) is a valid opening for σ_1
- (σ_2', π', f) is a valid opening for σ_1'

Then, $\sigma_2 = \sigma_2'$

Technical Tool: Projective Chainable Commitments



Intuitively: can associate CRS with an index j that allows projecting a commitment σ_1 onto a commitment to the first j indices

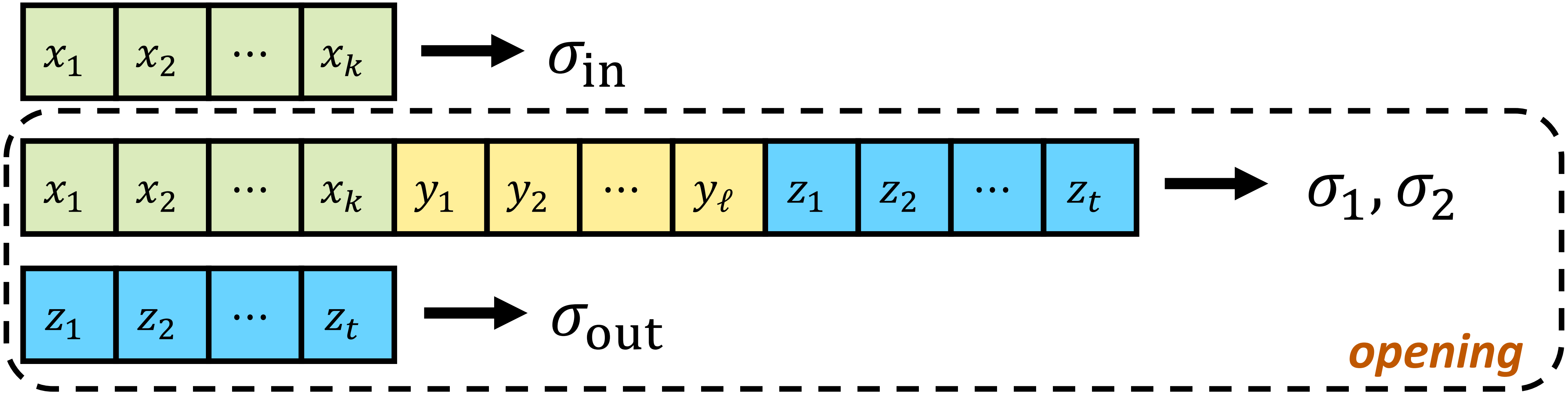
Projective chain binding: given $(\sigma_1, \sigma_2, \pi)$ and $(\sigma_1', \sigma_2', \pi')$

If $\mathrm{Project}(\mathrm{td}, \sigma_1, j) = \mathrm{Project}(\mathrm{td}, \sigma_1', j)$ and

- (σ_2, π, f) is a valid opening for σ_1
- (σ_2', π', f) is a valid opening for σ_1'

Then, $\mathrm{Project}(\mathrm{td}, \sigma_2, j+1) = \mathrm{Project}(\mathrm{td}, \sigma_2', j+1)$

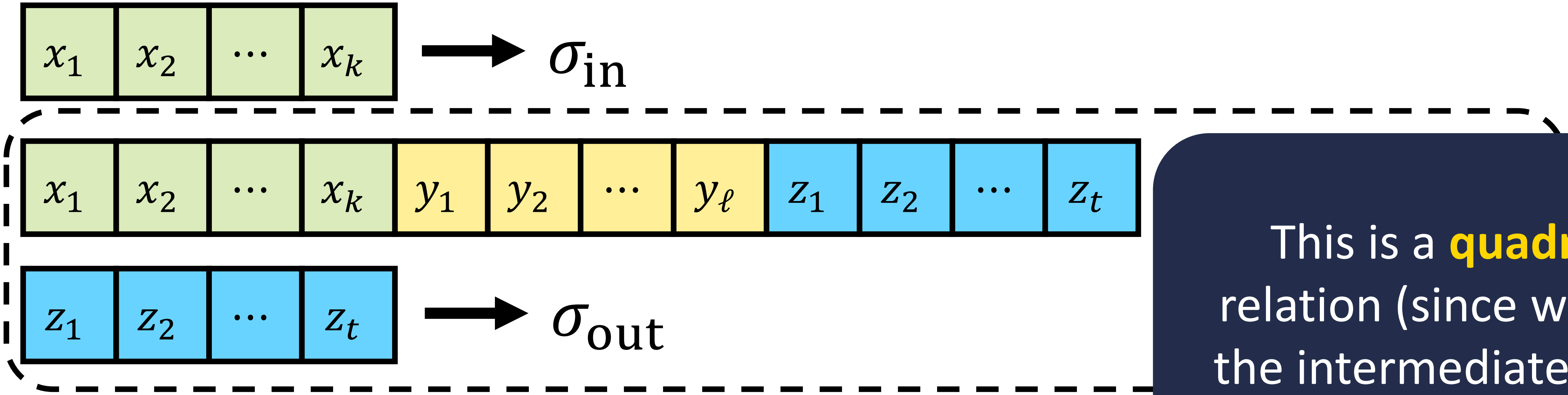
Using Projective Chainable Commitments



Prove statements of the following form:

- **Input consistency:** first k wires in σ_1 is consistent with σ_{input}
- **Gate consistency:** first $j + 1$ wires in σ_2 is consistent with first j wires in σ_1

Using Projective Chainable Commitments

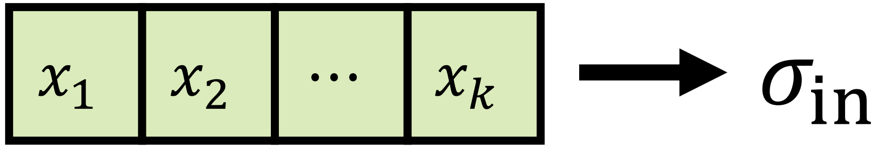


This is a **quadratic** relation (since we have the intermediate wires)

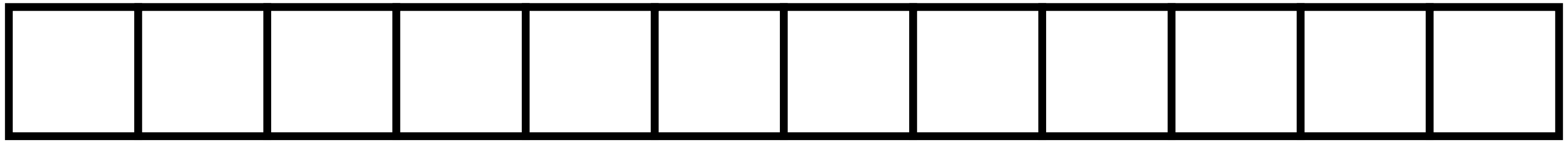
Prove statements of the following form:

- **Input consistency:** first k wires in σ_1 is consistent with σ_{input}
- **Gate consistency:** first $j+1$ wires in σ_2 is consistent with first j wires in σ_1
- **Internal consistency:** first j wires in σ_1 is consistent with first j wires in σ_2
- **Output consistency:** last t wires in σ_1 are consistent with σ_{output}

Using Projective Chainable Commitments

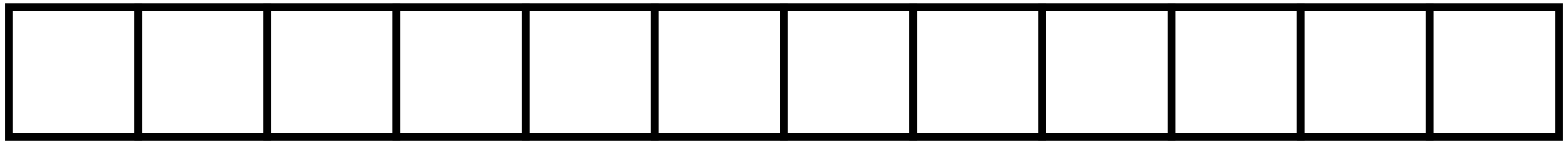


Consider two different openings: $(\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)$ and $(\sigma_1', \sigma_2', \sigma_{\text{out}}', \pi')$



σ_1, σ_1'

Initially: no guarantees on what $\sigma_1, \sigma_1', \sigma_2, \sigma_2'$ commit to

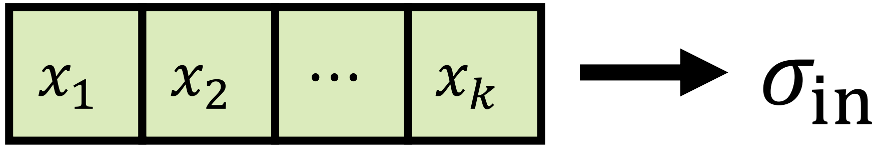


σ_2, σ_2'

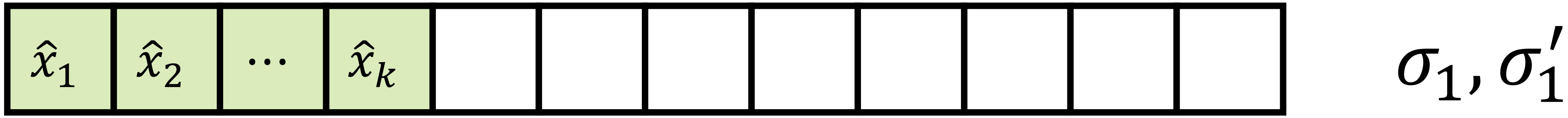
Step 1: Input consistency between σ_{in} and σ_1, σ_1'

Projective chain binding: σ_1, σ_1' are both openings for σ_{in} so $\text{Project}(\sigma_1, k) = \text{Project}(\sigma_1', k)$

Using Projective Chainable Commitments

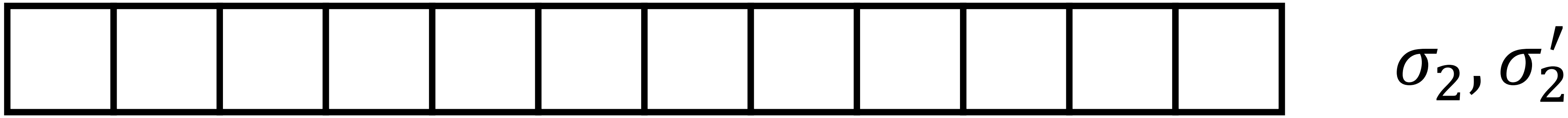


Consider two different openings: $(\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)$ and $(\sigma_1', \sigma_2', \sigma_{\text{out}}', \pi')$



σ_1 and σ_1' **agree** on first k components:
$\text{Project}(\sigma_1, k) = \text{Project}(\sigma_1', k)$

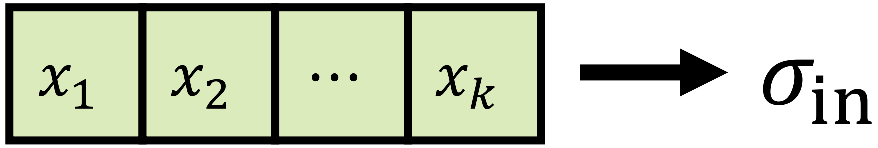
Note: we do **not** know what values they have, only that they agree



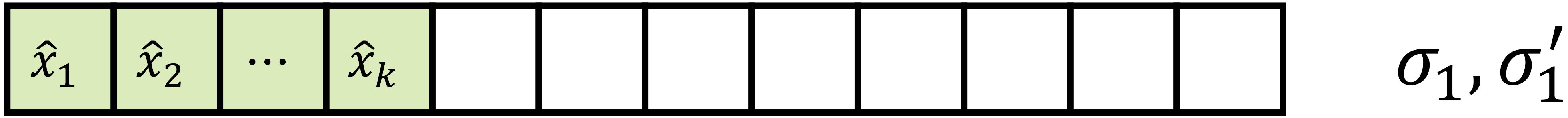
Step 1: Input consistency between σ_{in} and σ_1, σ_1'

Projective chain binding: σ_1, σ_1' are both openings for σ_{in} so $\text{Project}(\sigma_1, k) = \text{Project}(\sigma_1', k)$

Using Projective Chainable Commitments

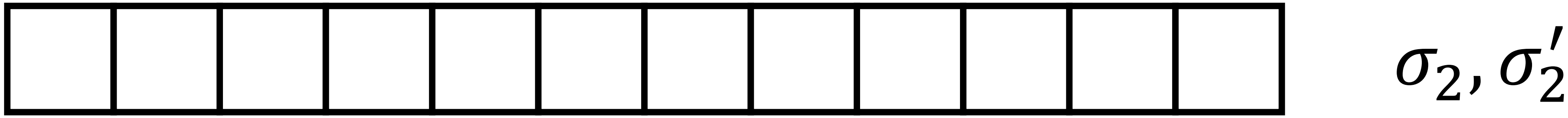


Consider two different openings: $(\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)$ and $(\sigma_1', \sigma_2', \sigma_{\text{out}}', \pi')$



σ_1 and σ_1' **agree** on first k components:
$\text{Project}(\sigma_1, k) = \text{Project}(\sigma_1', k)$

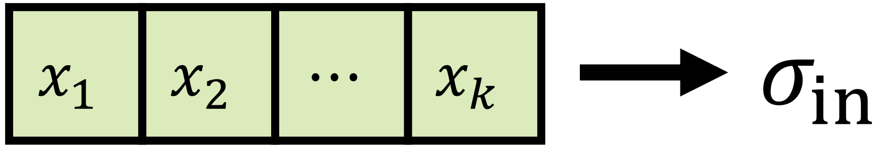
Note: we do **not** know what values they have, only that they agree



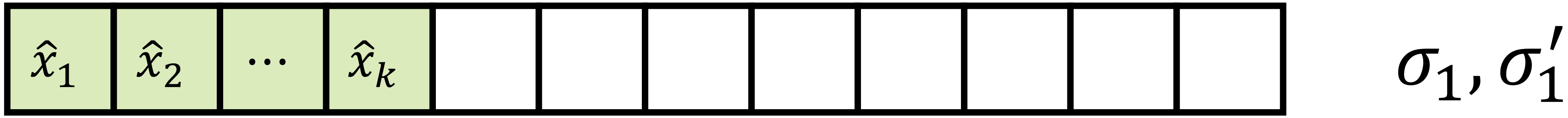
Step 2: Gate consistency between first k wires in σ_1, σ_1' with first $k+1$ wires in σ_2, σ_2'

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

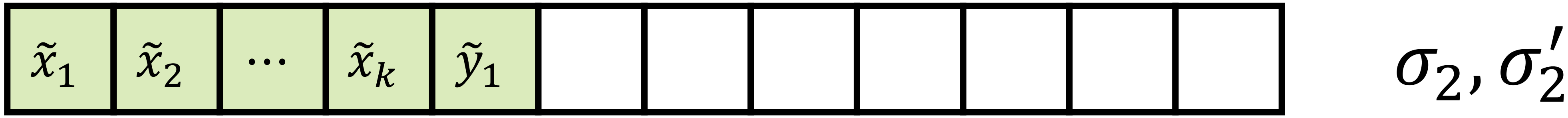
Using Projective Chainable Commitments



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



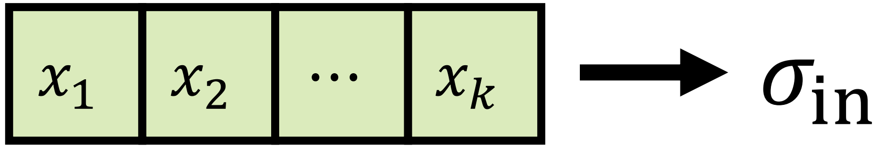
σ_2 and σ_2' agree on first $k+1$ components:
$\text{Project}(\sigma_2, k+1) = \text{Project}(\sigma_2', k+1)$



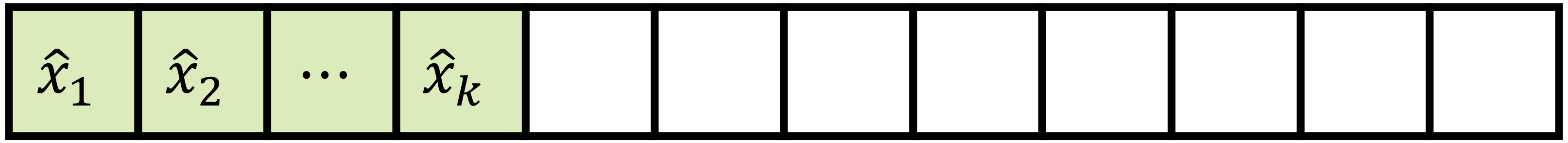
Step 2: Gate consistency between first k wires in σ_1, σ_1' with first $k+1$ wires in σ_2, σ_2'

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

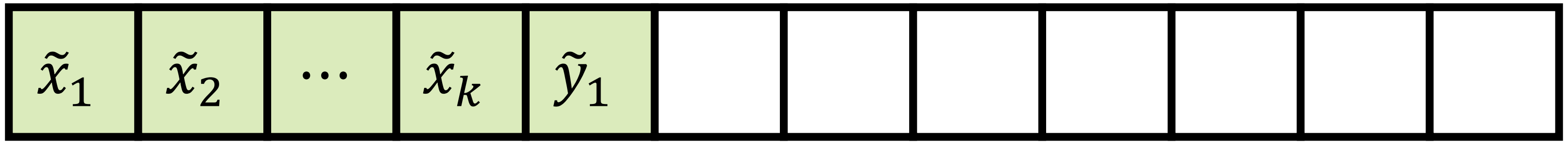


Consider two different openings: $(\sigma_1, \sigma_2, \sigma_{\text{out}}, \pi)$ and $(\sigma_1', \sigma_2', \sigma_{\text{out}}', \pi')$



σ_1, σ_1'

σ_2 and σ_2' agree on first $k+1$ components:
$\text{Project}(\sigma_2, k+1) = \text{Project}(\sigma_2', k+1)$

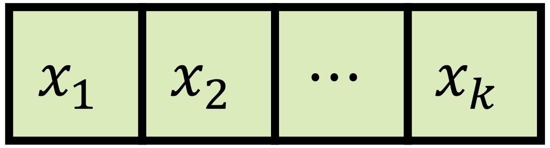


σ_2, σ_2'

Step 3: Internal consistency between first $k+1$ wires in σ_2, σ_2' with first $k+1$ wires in σ_1, σ_1'

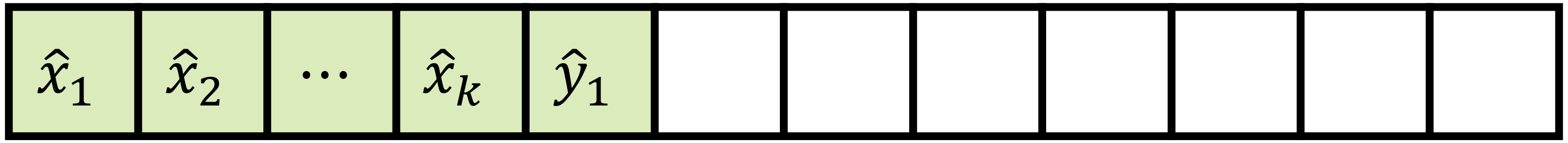
Since $\text{Project}(\sigma_2, k+1) = \text{Project}(\sigma_2', k+1)$, projective chain binding implies $\text{Project}(\sigma_1, k+1) = \text{Project}(\sigma_1', k+1)$

Using Projective Chainable Commitments



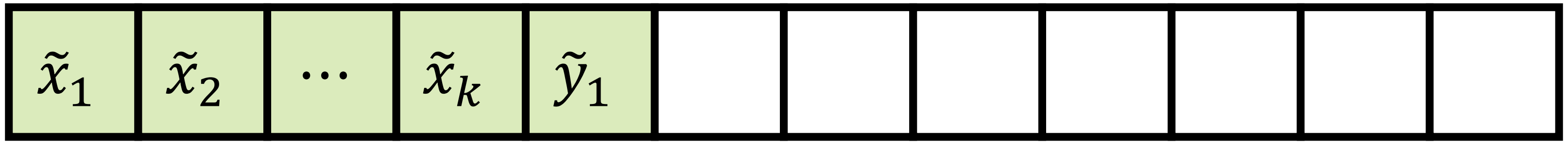
$\rightarrow \sigma_{\text{in}}$

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



σ_1, σ_1'

σ_1 and σ_1' agree on first $k+1$ components:
$\text{Project}(\sigma_1, k+1) = \text{Project}(\sigma_1', k+1)$

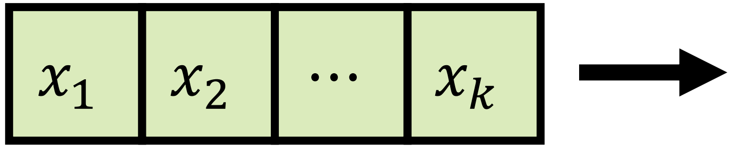


σ_2, σ_2'

Step 3: Internal consistency between first $k+1$ wires in σ_2, σ_2' with first $k+1$ wires in σ_1, σ_1'

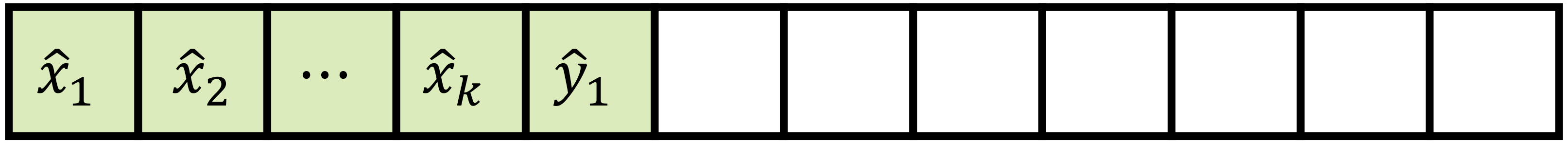
Since $\text{Project}(\sigma_2, k+1) = \text{Project}(\sigma_2', k+1)$, projective chain binding implies $\text{Project}(\sigma_1, k+1) = \text{Project}(\sigma_1', k+1)$

Using Projective Chainable Commitments



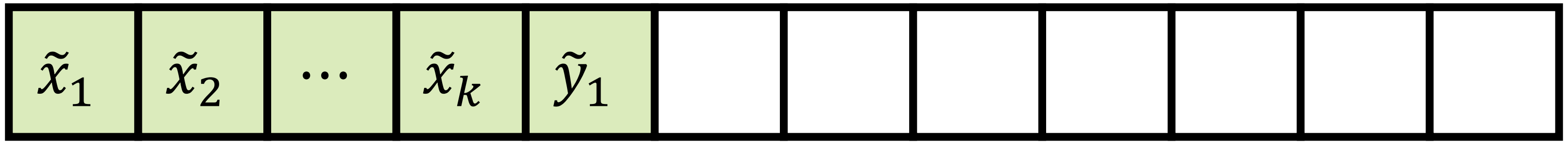
σ_{in}

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



σ_1, σ_1'

σ_1 and σ_1' agree on first $k+1$ components:
$\text{Project}(\sigma_1, k+1) = \text{Project}(\sigma_1', k+1)$



σ_2, σ_2'

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 σ_1, σ_1' agree on **all** components

Using Projective Chainable Commitments

x_1	x_2	$\cdots$	x_k

$\longrightarrow \sigma_{\text{in}}$

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

$\hat{x}_1$	$\hat{x}_2$	$\cdots$	$\hat{x}_k$	$\hat{y}_1$	$\hat{y}_2$	$\cdots$	$\hat{y}_\ell$	$\hat{z}_1$	$\hat{z}_2$	$\cdots$	$\hat{z}_t$

σ_1, σ_1'

$\tilde{x}_1$	$\tilde{x}_2$	$\cdots$	$\tilde{x}_k$	$\tilde{y}_1$	$\tilde{y}_2$	$\cdots$	$\tilde{y}_\ell$	$\tilde{z}_1$	$\tilde{z}_2$	$\cdots$	$\tilde{z}_\ell$

σ_2, σ_2'

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 σ_1, σ_1' agree on **all** components

Using Projective Chainable Commitments

x_1	x_2	$\cdots$	x_k

$\rightarrow \sigma_{\text{in}}$

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

$\hat{x}_1$	$\hat{x}_2$	$\cdots$	$\hat{x}_k$	$\hat{y}_1$	$\hat{y}_2$	$\cdots$	$\hat{y}_\ell$	$\hat{z}_1$	$\hat{z}_2$	$\cdots$	$\hat{z}_t$

σ_1, σ_1'

$\tilde{x}_1$	$\tilde{x}_2$	$\cdots$	$\tilde{x}_k$	$\tilde{y}_1$	$\tilde{y}_2$	$\cdots$	$\tilde{y}_\ell$	$\tilde{z}_1$	$\tilde{z}_2$	$\cdots$	$\tilde{z}_\ell$

σ_2, σ_2'

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]

Summary

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

Scheme	Function Class	$\lvert\mathrm{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\pi\rvert$	Assumption
This work	arithmetic circuits	$\boldsymbol{O(s^5)}$	$O(1)$	$O(1)$	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 $\mathrm{poly}(\lambda)$ 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