Polynomial Commitments from Lattices:

Post-Quantum Security, Fast Verification and Transparent Setup

Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, Hoeteck Wee

NTT Research Bocconi King's College London

Motivations: SNARKs

SNARK = Succint Non-interactive ARgument of Knowledge

Motivations: SNARKs

SNARK = Succint Non-interactive ARgument of Knowledge

verifiable computation

multi-party computation

anonymous credentials

blockchain

......

Constructing SNARKs

Modular approach

Constructing SNARKs

Modular approach

information theoretic
component

Polynomial IOP

Constructing SNARKs

Modular approach

information theoretic component

Polynomial IOP

+

Constructing SNARKs

Modular approach

information theoretic component	+	cryptographic component
Polynomial IOP		Polynomial CS

Constructing SNARKs

Modular approach

information theoretic component

Polynomial IOP

$+$

cryptographic component

Polynomial CS

focus of these works!

Result

Result

Polynomial CS with bunch of nice properties:

Result

Polynomial CS with bunch of nice properties:

- quasi-linear prover time
- transparent setup
- succinct commitment
- fast verification
- binding under standard assumptions (SIS)
- post-quantum security

Result

Polynomial CS with bunch of nice properties:

- quasi-linear prover time
- transparent setup
- succinct commitment
- fast verification
- binding under standard assumptions (SIS)
- post-quantum security

Concrete Efficiency!

2^{15}	2^{20}	2^{25}	2^{30}
91KB	**403**KB	**1.36**MB	**4.90**MB

Polynomial Commitment Scheme (PCS)

Polynomial Commitment Scheme (PCS)

Commitment Scheme

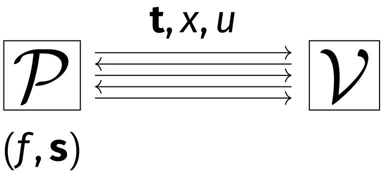
- commitment to vector $\mathbf{f} \in \mathbb{Z}_q^d$
- commitment $\mathbf{t}$, opening $\mathbf{s}$

Polynomial Commitment Scheme (PCS)

Commitment Scheme

- commitment to vector $\mathbf{f} \in \mathbb{Z}_q^d$
- commitment $\mathbf{t}$, opening $\mathbf{s}$

Evaluation Protocol



"Knows f such that $f(x) = u$ and opening $\mathbf{s}$ for $\mathbf{f} = \mathbf{coeff}(f), \mathbf{t}$"

Polynomial Commitment Scheme (PCS)

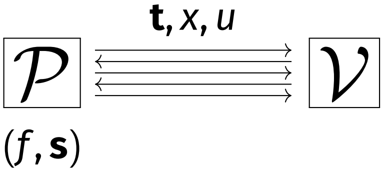
Commitment Scheme

- commitment to vector $\mathbf{f} \in \mathbb{Z}_q^d$
- commitment $\mathbf{t}$, opening $\mathbf{s}$

succinct commitment: $|\mathbf{t}| \ll d$

vectors $\mathbf{f}$ of arbitrary norm

Evaluation Protocol



"Knows f such that $f(x) = u$ and opening $\mathbf{s}$ for $\mathbf{f} = \mathbf{coeff}(f), \mathbf{t}$"

Polynomial Commitment Scheme (PCS)

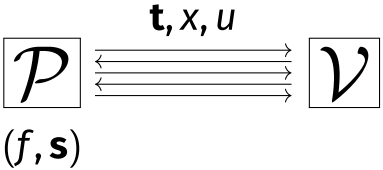
Commitment Scheme

- commitment to vector $\mathbf{f} \in \mathbb{Z}_q^d$
- commitment $\mathbf{t}$, opening $\mathbf{s}$

succinct commitment: $|\mathbf{t}| \ll d$

vectors $\mathbf{f}$ of arbitrary norm

Evaluation Protocol



"Knows f such that $f(x) = u$ and opening $\mathbf{s}$ for $\mathbf{f} = \mathbf{coeff}(f), \mathbf{t}$"

$\mathcal{V}$'s running time $\ll d$

How to Commit: from 2n to n

How to Commit: from 2n to n

crs : $\mathbf{A} \in \mathbb{Z}_q^{n \times 2n \log q}$, $\quad \mathbf{G} \in \mathbb{Z}_q^{2n \times 2n \log q}$

How to Commit: from 2n to n

$$\mathsf{crs}: \ \mathbf{A} \in \mathbb{Z}_q^{n \times 2n \log q}, \quad \mathbf{G} \in \mathbb{Z}_q^{2n \times 2n \log q} \qquad \text{(2 generalizes to } r\text{)}$$

How to Commit: from 2n to n

$\mathsf{crs}: \ \mathbf{A} \in \mathbb{Z}_q^{n \times 2n \log q}, \quad \mathbf{G} \in \mathbb{Z}_q^{2n \times 2n \log q}$ (2 generalizes to r)

To commit to $\mathbf{f} \in \mathbb{Z}_q^{2n}$ compute $\mathbf{t} = H(\mathbf{f})$

How to Commit: from 2n to n

crs : $\mathbf{A} \in \mathbb{Z}_q^{n \times 2n \log q}$, $\quad \mathbf{G} \in \mathbb{Z}_q^{2n \times 2n \log q}$ $\quad$ (2 generalizes to r)

To commit to $\mathbf{f} \in \mathbb{Z}_q^{2n}$ compute $\mathbf{t} = H(\mathbf{f})$

$$\begin{aligned} H : \mathbb{Z}_q^{2n} &\longrightarrow \mathbb{Z}_q^{n} \\ \mathbf{f} &\longmapsto \mathbf{A} \cdot \mathbf{G}^{-1}(\mathbf{f}) \end{aligned}$$

How to Commit: from 2n to n

$\mathsf{crs}: \ \mathbf{A} \in \mathbb{Z}_q^{n \times 2n \log q}, \quad \mathbf{G} \in \mathbb{Z}_q^{2n \times 2n \log q}$ (2 generalizes to r)

To commit to $\mathbf{f} \in \mathbb{Z}_q^{2n}$ compute $\mathbf{t} = H(\mathbf{f})$

$$\begin{aligned} H : \mathbb{Z}_q^{2n} &\longrightarrow \mathbb{Z}_q^{n} \\ \mathbf{f} &\longmapsto \mathbf{A} \cdot \mathbf{G}^{-1}(\mathbf{f}) \end{aligned}$$

To open provide low-norm $\mathbf{s} \in \mathbb{Z}^{2n \log q}$

$$\mathbf{A} \cdot \mathbf{s} = \mathbf{t} \mod q$$

$$\mathbf{G} \cdot \mathbf{s} = \mathbf{f} \mod q$$

Tree-based Approach

Want to commit to $\mathbf{f} \in \mathbb{Z}_q^{8n}$

Tree-based Approach

Want to commit to $\mathbf{f} \in \mathbb{Z}_q^{8n}$

$\mathbf{f}$: $\boxed{\mathbf{f}_{000} \quad \mathbf{f}_{001} \quad \mathbf{f}_{010} \quad \mathbf{f}_{011} \quad \mathbf{f}_{100} \quad \mathbf{f}_{101} \quad \mathbf{f}_{110} \quad \mathbf{f}_{111}}$

Tree-based Approach

Want to commit to $\mathbf{f} \in \mathbb{Z}_q^{8n}$

$\mathbf{t}_{00}$

$\mathbf{f}: \boxed{\mathbf{f}_{000} \quad \mathbf{f}_{001} \quad \mathbf{f}_{010} \quad \mathbf{f}_{011} \quad \mathbf{f}_{100} \quad \mathbf{f}_{101} \quad \mathbf{f}_{110} \quad \mathbf{f}_{111}}$

Tree-based Approach

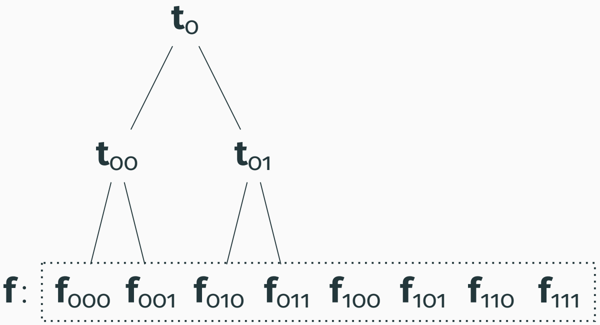
Want to commit to $\mathbf{f} \in \mathbb{Z}_q^{8n}$

$\mathbf{t}_{00}$ $\mathbf{t}_{01}$

$\mathbf{f}$: $\mathbf{f}_{000}$ $\mathbf{f}_{001}$ $\mathbf{f}_{010}$ $\mathbf{f}_{011}$ $\mathbf{f}_{100}$ $\mathbf{f}_{101}$ $\mathbf{f}_{110}$ $\mathbf{f}_{111}$

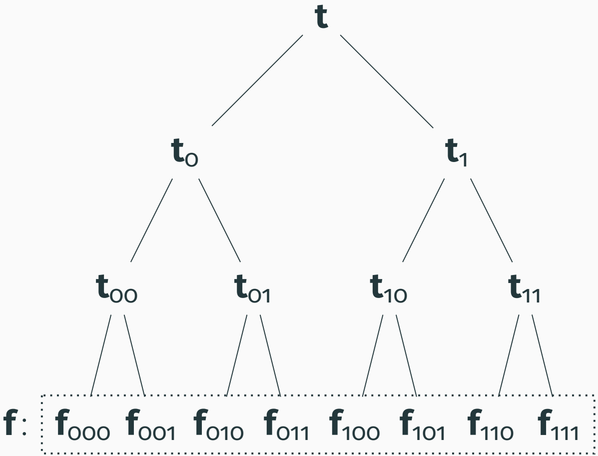
Tree-based Approach

Want to commit to $\mathbf{f} \in \mathbb{Z}_q^{8n}$



Tree-based Approach

Want to commit to $\mathbf{f} \in \mathbb{Z}_q^{8n}$



Opening

Algebraic Viewpoint

$$\mathbf{t} = (\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{G}^{-1}\left((\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{G}^{-1}\Big((\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{G}^{-1}(\mathbf{f})\Big)\right)$$

Opening

Algebraic Viewpoint

$$\mathbf{t} = (\mathbf{I} \otimes \mathbf{A}) \cdot \underbrace{\mathbf{G}^{-1}\left((\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{G}^{-1}\left((\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{G}^{-1}(\mathbf{f}) \right) \right)}_{\mathbf{s}_0}$$

Opening

Algebraic Viewpoint

$$\mathbf{t} = (\mathbf{I} \otimes \mathbf{A}) \cdot \underbrace{\mathbf{G}^{-1}\left((\mathbf{I} \otimes \mathbf{A}) \cdot \overbrace{\mathbf{G}^{-1}\Big((\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{G}^{-1}(\mathbf{f}) \Big)}^{\mathbf{s}_1} \right)}_{\mathbf{s}_0}$$

Opening

Algebraic Viewpoint

$$\mathbf{t} = (\mathbf{I} \otimes \mathbf{A}) \cdot \underbrace{\mathbf{G}^{-1}\left((\mathbf{I} \otimes \mathbf{A}) \cdot \overbrace{\mathbf{G}^{-1}\Big((\mathbf{I} \otimes \mathbf{A}) \cdot \underbrace{\mathbf{G}^{-1}(\mathbf{f})}_{\mathbf{s}_2} \Big)}^{\mathbf{s}_1} \right)}_{\mathbf{s}_0}$$

Opening

Algebraic Viewpoint

$$\mathbf{t} = (\mathbf{I} \otimes \mathbf{A}) \cdot \underbrace{\mathbf{G}^{-1}\left((\mathbf{I} \otimes \mathbf{A}) \cdot \overbrace{\mathbf{G}^{-1}\Big((\mathbf{I} \otimes \mathbf{A}) \cdot \underbrace{\mathbf{G}^{-1}(\mathbf{f})}_{\mathbf{s}_2} \Big)}^{\mathbf{s}_1} \right)}_{\mathbf{s}_0}$$

Opening: "short" $\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2)$ such that

$$\mathbf{A} \cdot \mathbf{s}_0 = \mathbf{t}$$
$$\mathbf{G} \cdot \mathbf{s}_0 = (\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{s}_1 \quad \text{and} \quad \mathbf{G} \cdot \mathbf{s}_1 = (\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{s}_2$$
$$\mathbf{G} \cdot \mathbf{s}_2 = \mathbf{f}$$

Opening

Algebraic Viewpoint

$$\mathbf{t} = (\mathbf{I} \otimes \mathbf{A}) \cdot \underbrace{\mathbf{G}^{-1}\left((\mathbf{I} \otimes \mathbf{A}) \cdot \overbrace{\mathbf{G}^{-1}\Big((\mathbf{I} \otimes \mathbf{A}) \cdot \underbrace{\mathbf{G}^{-1}(\mathbf{f})}_{\mathbf{s}_2} \Big)}^{\mathbf{s}_1} \right)}_{\mathbf{s}_0}$$

Opening: "short" $\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2)$ such that

$$\boxed{\begin{array}{l} \mathbf{A} \cdot \mathbf{s}_0 = \mathbf{t} \\ \mathbf{G} \cdot \mathbf{s}_0 = (\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{s}_1 \quad \text{and} \quad \mathbf{G} \cdot \mathbf{s}_1 = (\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{s}_2 \\ \mathbf{G} \cdot \mathbf{s}_2 = \mathbf{f} \end{array}}$$

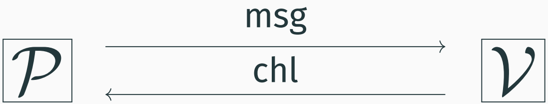
opening relation

Folding Friendly I

Reduction (of Knowledge) framework [KP23].

Folding Friendly I

Reduction (of Knowledge) framework [KP23].



Folding Friendly I

Reduction (of Knowledge) framework [KP23].

$$\boxed{\mathcal{P}} \quad \begin{array}{c} \xrightarrow{\quad\quad\text{msg}\quad\quad} \\ \xleftarrow[\quad\quad\text{chl}\quad\quad]{} \end{array} \quad \boxed{\mathcal{V}}$$

commitment	$\mathbf{t}$
vector	$\mathbf{f}$
opening	$\mathbf{s}$

Folding Friendly I

Reduction (of Knowledge) framework [KP23].

$$\boxed{\mathcal{P}} \quad \begin{array}{c} \xrightarrow{\quad\quad\textsf{msg}\quad\quad} \\ \xleftarrow{\quad\quad\textsf{chl}\quad\quad} \end{array} \quad \boxed{\mathcal{V}}$$

message **msg** + folding challenge **chl**

commitment **t**

vector **f**

opening **s**

Folding Friendly I

Reduction (of Knowledge) framework [KP23].

$$\boxed{\mathcal{P}} \quad \underset{\xleftarrow[]{}}{\xrightarrow{\quad\textsf{msg}\quad}} \!\!\!\!\!\!\!\!\!\!\!\!\raisebox{-0.6em}{\textsf{chl}} \quad \boxed{\mathcal{V}}$$

message $\textbf{msg}$ + folding challenge $\textbf{chl}$

$\downarrow$

commitment	$\mathbf{t}$	$\longrightarrow$	$\mathbf{t}'$
vector	$\mathbf{f}$	$\longrightarrow$	$\mathbf{f}'$
opening	$\mathbf{s}$		$\mathbf{s}'$

Folding Friendly I

Reduction (of Knowledge) framework [KP23].

$$\boxed{\mathcal{P}} \quad \begin{array}{c} \textbf{msg} \\ \xrightarrow{\hspace{6em}} \\ \textbf{chl} \\ \xleftarrow{\hspace{6em}} \end{array} \quad \boxed{\mathcal{V}}$$

message **msg** + folding challenge **chl**

$$\downarrow$$

commitment $\mathbf{t} \longrightarrow \mathbf{t}'$

vector $\mathbf{f}$, opening $\mathbf{s}$ $\longrightarrow$ $\mathbf{f}'$, $\mathbf{s}'$

$$|\mathbf{f}'| = |\mathbf{f}|/2$$

$$|\mathbf{s}'| \approx |\mathbf{s}|/2$$

Folding Friendly II

$\mathbf{A}, \mathbf{t}$

$\boxed{\mathcal{P}}$

$\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2), \mathbf{f}$

$\boxed{\mathcal{V}}$

Folding Friendly II

$$\boxed{\mathbf{A}, \mathbf{t}}$$

$$\boxed{\mathcal{P}} \xrightarrow{\text{msg} = \mathbf{s}_0} \boxed{\mathcal{V}}$$

$\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2), \mathbf{f}$

Folding Friendly II

$$\boxed{\mathbf{A}, \mathbf{t}}$$

$$\boxed{\mathcal{P}} \xrightarrow{\text{msg} = \mathbf{s}_0} \boxed{\mathcal{V}} \qquad (\mathbf{I} \otimes \mathbf{A}) \cdot \text{msg} \stackrel{?}{=} \mathbf{t}$$

$$\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2), \mathbf{f}$$

Folding Friendly II

$$\boxed{\mathbf{A}, \mathbf{t}}$$

$$\boxed{\mathcal{P}} \quad \xrightarrow{\text{msg} = \mathbf{s}_0} \quad \boxed{\mathcal{V}} \qquad (\mathbf{I} \otimes \mathbf{A}) \cdot \text{msg} \stackrel{?}{=} \mathbf{t}$$

$$\xleftarrow{\text{chl} = \mathbf{c} \in \{0,1\}^2}$$

$$\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2), \mathbf{f}$$

Folding Friendly II

$$\mathbf{A}, \mathbf{t}$$

building block

$$\boxed{\mathcal{P}} \quad \xrightarrow{\text{msg} = \mathbf{s}_0} \quad \boxed{\mathcal{V}} \qquad (\mathbf{I} \otimes \mathbf{A}) \cdot \text{msg} \stackrel{?}{=} \mathbf{t}$$

$$\xleftarrow{\text{chl} = \mathbf{c} \in \{0,1\}^2}$$

$$\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2), \mathbf{f}$$

Folding Friendly II

$$\boxed{\mathbf{A}, \mathbf{t}}$$

building block

$$\boxed{\mathcal{P}} \underset{\xleftarrow{}}{\xrightarrow[\text{chl} = \mathbf{c} \in \{0,1\}^2]{\text{msg} = \mathbf{s}_0}} \boxed{\mathcal{V}} \qquad (\mathbf{I} \otimes \mathbf{A}) \cdot \text{msg} \stackrel{?}{=} \mathbf{t}$$

$$\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2), \mathbf{f}$$

$$\mathbf{t}' := (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{G} \cdot \text{msg}$$

Folding Friendly II

$$\boxed{\mathbf{A}, \mathbf{t}}$$

building block

$$\boxed{\mathcal{P}} \quad \xrightarrow{\text{msg} = \mathbf{s}_0} \quad \boxed{\mathcal{V}} \qquad (\mathbf{I} \otimes \mathbf{A}) \cdot \text{msg} \stackrel{?}{=} \mathbf{t}$$

$$\boxed{\mathcal{P}} \quad \xleftarrow{\text{chl} = \mathbf{c} \in \{0,1\}^2} \quad \boxed{\mathcal{V}}$$

$$\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2), \mathbf{f}$$

$$\mathbf{t}' := (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{G} \cdot \text{msg}$$
$$\mathbf{f}' := (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{f}$$
$$\mathbf{s}_0' := (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{s}_1$$
$$\mathbf{s}_1' := (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{s}_2$$

Folding Friendly II

$$\boxed{\mathbf{A}, \mathbf{t}}$$

building block

$$\boxed{\mathcal{P}} \quad \begin{array}{c} \xrightarrow{\text{msg} = \mathbf{s}_0} \\ \xleftarrow[\text{chl} = \mathbf{c} \in \{0,1\}^2]{} \end{array} \quad \boxed{\mathcal{V}} \qquad (\mathbf{I} \otimes \mathbf{A}) \cdot \text{msg} \stackrel{?}{=} \mathbf{t}$$

$$\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2), \mathbf{f}$$

$$\begin{aligned} \mathbf{t}' &:= (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{G} \cdot \text{msg} \\ \mathbf{f}' &:= (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{f} \\ \mathbf{s}'_0 &:= (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{s}_1 \\ \mathbf{s}'_1 &:= (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{s}_2 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \mathbf{A} \cdot \mathbf{s}'_0 &= \mathbf{t}' \\ \mathbf{G} \cdot \mathbf{s}'_0 &= (\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{s}'_1 \\ \mathbf{G} \cdot \mathbf{s}'_1 &= \mathbf{f}' \end{aligned}$$

Folding Friendly II

$$\boxed{\mathbf{A}, \mathbf{t}}$$

building block

$$\boxed{\mathcal{P}} \underset{\xleftarrow[]{}}{\xrightarrow[\mathrm{chl} = \mathbf{c} \in \{0,1\}^2]{\mathrm{msg} = \mathbf{s}_0}} \boxed{\mathcal{V}} \qquad (\mathbf{I} \otimes \mathbf{A}) \cdot \mathrm{msg} \stackrel{?}{=} \mathbf{t}$$

$$\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2), \mathbf{f}$$

$$\begin{aligned} \mathbf{t}' &:= (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{G} \cdot \mathrm{msg} \\ \mathbf{f}' &:= (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{f} \\ \mathbf{s}_0' &:= (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{s}_1 \\ \mathbf{s}_1' &:= (\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{s}_2 \end{aligned} \implies \boxed{\begin{aligned} \mathbf{A} \cdot \mathbf{s}_0' &= \mathbf{t}' \\ \mathbf{G} \cdot \mathbf{s}_0' &= (\mathbf{I} \otimes \mathbf{A}) \cdot \mathbf{s}_1' \\ \mathbf{G} \cdot \mathbf{s}_1' &= \mathbf{f}' \end{aligned}}$$

opening relation

Knowledge Soundness I

Knowledge Soundness I

Coordinate-Wise extraction strategy from [BBCdGL18; FMN23]:

Knowledge Soundness I

Coordinate-Wise extraction strategy from [BBCdGL18; FMN23]:

Rewind cheating prover to obtain

$$\text{openings } \mathbf{s}', \mathbf{s}'', \mathbf{s}''' \text{ for challenges } \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2 \text{ s.t.}$$

Knowledge Soundness I

Coordinate-Wise extraction strategy from [BBCdGL18; FMN23]:

Rewind cheating prover to obtain

openings $\mathbf{s}', \mathbf{s}'', \mathbf{s}'''$ for challenges $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ s.t.

$\boxed{\mathbf{c}_i \text{ agrees with } \mathbf{c}_0 \text{ in all rows except row } \boldsymbol{i}}$

Knowledge Soundness I

Coordinate-Wise extraction strategy from [BBCdGL18; FMN23]:

Rewind cheating prover to obtain

openings $\mathbf{s}', \mathbf{s}'', \mathbf{s}'''$ for challenges $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ s.t.

$\boxed{\mathbf{c}_i \text{ agrees with } \mathbf{c}_0 \text{ in all rows except row } \boldsymbol{i}}$

$\boxed{\mathcal{P}^*}$ $\boxed{\mathcal{V}}$

Knowledge Soundness I

Coordinate-Wise extraction strategy from [BBCdGL18; FMN23]:

Rewind cheating prover to obtain

openings $\mathbf{s}', \mathbf{s}'', \mathbf{s}'''$ for challenges $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ s.t.

$\boxed{\mathbf{c}_i \text{ agrees with } \mathbf{c}_0 \text{ in all rows except row } \boldsymbol{i}}$

$$\boxed{\mathcal{P}^*} \xrightarrow{\text{msg}} \boxed{\mathcal{V}}$$

Knowledge Soundness I

Coordinate-Wise extraction strategy from [BBCdGL18; FMN23]:

Rewind cheating prover to obtain

openings $\mathbf{s}', \mathbf{s}'', \mathbf{s}'''$ for challenges $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ s.t.

$$\boxed{\mathbf{c}_i \text{ agrees with } \mathbf{c}_0 \text{ in all rows except row } \boldsymbol{i}}$$

$$\boxed{\mathcal{P}^*} \quad \begin{array}{c} \xrightarrow{\qquad\quad \text{msg} \qquad\quad} \\ \xleftarrow[\qquad\qquad\qquad]{\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2} \end{array} \quad \boxed{\mathcal{V}}$$

Knowledge Soundness I

Coordinate-Wise extraction strategy from [BBCdGL18; FMN23]:

Rewind cheating prover to obtain

openings $\mathbf{s}', \mathbf{s}'', \mathbf{s}'''$ for challenges $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ s.t.

$\boxed{\mathbf{c}_i \text{ agrees with } \mathbf{c}_0 \text{ in all rows except row } \boldsymbol{i}}$

$$\boxed{\mathcal{P}^*} \quad \underset{\xleftarrow[\quad\quad\quad\quad\quad]{}}{\xrightarrow[\mathbf{c}_0,\mathbf{c}_1,\mathbf{c}_2]{\text{msg}}} \quad \boxed{\mathcal{V}}$$

$\mathbf{t}'$	$\mathbf{t}''$	$\mathbf{t}'''$
$\mathbf{s}', \mathbf{f}'$	$\mathbf{s}'', \mathbf{f}''$	$\mathbf{s}''', \mathbf{f}'''$

Knowledge Soundness I

Coordinate-Wise extraction strategy from [BBCdGL18; FMN23]:

Rewind cheating prover to obtain

openings $\mathbf{s}', \mathbf{s}'', \mathbf{s}'''$ for challenges $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ s.t.

$\boxed{\mathbf{c}_i \text{ agrees with } \mathbf{c}_0 \text{ in all rows except row } \boldsymbol{i}}$

$$\boxed{\mathcal{P}^*} \quad \underset{\xleftarrow[]{\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2}}{\xrightarrow{\text{msg}}} \quad \boxed{\mathcal{V}}$$

$$\begin{array}{ccc} \mathbf{t}' & \mathbf{t}'' & \mathbf{t}''' \end{array}$$

$$\underbrace{\mathbf{s}', \mathbf{f}' \qquad \mathbf{s}'', \mathbf{f}'' \qquad \mathbf{s}''', \mathbf{f}'''}_{\text{How to recover opening of } \mathbf{t}?}$$

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{c}_0 - \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{c}_0 - \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \implies \begin{bmatrix} \mathbf{c}_0 \middle| \mathbf{c}_1 \middle| \mathbf{c}_2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{c}_0 - \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \implies \begin{bmatrix} \mathbf{c}_0 | \mathbf{c}_1 | \mathbf{c}_2 \end{bmatrix} \cdot \overbrace{\begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}}^{\mathbf{H}}$$

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{c}_0 - \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \implies \begin{bmatrix} \mathbf{c}_0 | \mathbf{c}_1 | \mathbf{c}_2 \end{bmatrix} \cdot \overbrace{\begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}}^{\mathbf{H}} = \mathbf{I}$$

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{c}_0 - \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \implies \begin{bmatrix} \mathbf{c}_0 | \mathbf{c}_1 | \mathbf{c}_2 \end{bmatrix} \cdot \overbrace{\begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}}^{\mathbf{H}} = \mathbf{I}$$

Use **H** to "invert" folding!

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{c}_0 - \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \implies \begin{bmatrix} \mathbf{c}_0 | \mathbf{c}_1 | \mathbf{c}_2 \end{bmatrix} \cdot \overbrace{\begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}}^{\mathbf{H}} = \mathbf{I}$$

Use **H** to "invert" folding!

Challenge space has small size

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{c}_0 - \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \implies \begin{bmatrix} \mathbf{c}_0 | \mathbf{c}_1 | \mathbf{c}_2 \end{bmatrix} \cdot \overbrace{\begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}}^{\mathbf{H}} = \mathbf{I}$$

Use **H** to "invert" folding!

Challenge space has small size: parallel repetition?

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{c}_0 - \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \implies \begin{bmatrix} \mathbf{c}_0 | \mathbf{c}_1 | \mathbf{c}_2 \end{bmatrix} \cdot \overbrace{\begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}}^{\mathbf{H}} = \mathbf{I}$$

Use $\mathbf{H}$ to "invert" folding!

Challenge space has small size: parallel repetition? ⚠

Knowledge Soundness II

How to recover opening $\mathbf{s}^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*), \mathbf{f}^*$ of $\mathbf{t}$?

$$\mathbf{c}_0 - \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{c}_0 - \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \implies \begin{bmatrix} \mathbf{c}_0 | \mathbf{c}_1 | \mathbf{c}_2 \end{bmatrix} \cdot \overbrace{\begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}}^{\mathbf{H}} = \mathbf{I}$$

Use $\mathbf{H}$ to "invert" folding!

Challenge space has small size: parallel repetition?

To achieve negligible soundness error: $\mathbf{chl} = \mathbf{C} \leftarrow \{0,1\}^{\kappa \cdot 2 \times \kappa}$.

Polynomial Evaluation

Ex: $f \in \mathbb{Z}[X]$ polynomial of degree < 8, $\mathsf{coeff}(f) = \mathbf{f} \in \mathbb{Z}^8$.

Polynomial Evaluation

Ex: $f \in \mathbb{Z}[X]$ polynomial of degree < 8, $\text{coeff}(f) = \mathbf{f} \in \mathbb{Z}^8$.

$$f(x) = u$$

Polynomial Evaluation

Ex: $f \in \mathbb{Z}[X]$ polynomial of degree < 8, $\mathrm{coeff}(f) = \mathbf{f} \in \mathbb{Z}^8$.

$$f(x) = u \iff (\mathbf{I} \otimes \mathbf{x}_2) \cdot (\mathbf{I} \otimes \mathbf{x}_1) \cdot (\mathbf{I} \otimes \mathbf{x}_0) \cdot \mathbf{f} = u$$

with $\mathbf{x}_i = \left[\mathbf{1},\, x^{2^i}\right]$.

Polynomial Evaluation

Ex: $f \in \mathbb{Z}[X]$ polynomial of degree < 8, $\mathrm{coeff}(f) = \mathbf{f} \in \mathbb{Z}^8$.

$$f(x) = u \iff (\mathbf{I} \otimes \mathbf{x}_2) \cdot (\mathbf{I} \otimes \mathbf{x}_1) \cdot (\mathbf{I} \otimes \mathbf{x}_0) \cdot \mathbf{f} = u$$

with $\mathbf{x}_i = \left[\mathbf{1}, x^{2^i}\right]$.

$$(\mathbf{I} \otimes \mathbf{x}_2) \cdot (\mathbf{I} \otimes \mathbf{x}_1) \cdot (\mathbf{I} \otimes \mathbf{x}_0) \cdot \mathbf{f} = u$$

Polynomial Evaluation

Ex: $f \in \mathbb{Z}[X]$ polynomial of degree < 8, $\text{coeff}(f) = \mathbf{f} \in \mathbb{Z}^8$.

$$f(x) = u \iff (\mathbf{I} \otimes \mathbf{x}_2) \cdot (\mathbf{I} \otimes \mathbf{x}_1) \cdot (\mathbf{I} \otimes \mathbf{x}_0) \cdot \mathbf{f} = u$$

with $\mathbf{x}_i = \left[\mathbf{1}, x^{2^i}\right]$.

$$(\mathbf{I} \otimes \mathbf{x}_2) \cdot \underbrace{(\mathbf{I} \otimes \mathbf{x}_1) \cdot (\mathbf{I} \otimes \mathbf{x}_0) \cdot \mathbf{f}}_{\mathbf{v}} = u$$

Polynomial Evaluation

Ex: $f \in \mathbb{Z}[X]$ polynomial of degree < 8, $\mathrm{coeff}(f) = \mathbf{f} \in \mathbb{Z}^8$.

$$f(x) = u \iff (\mathbf{I} \otimes \mathbf{x}_2) \cdot (\mathbf{I} \otimes \mathbf{x}_1) \cdot (\mathbf{I} \otimes \mathbf{x}_0) \cdot \mathbf{f} = u$$

with $\mathbf{x}_i = \begin{bmatrix} 1, & x^{2^i} \end{bmatrix}$.

$$(\mathbf{I} \otimes \mathbf{x}_2) \cdot \underbrace{(\mathbf{I} \otimes \mathbf{x}_1) \cdot (\mathbf{I} \otimes \mathbf{x}_0) \cdot \mathbf{f}}_{\mathbf{v}} = u$$

$$\Downarrow$$

$$(\mathbf{I} \otimes \mathbf{x}_1) \cdot (\mathbf{I} \otimes \mathbf{x}_0) \cdot \underbrace{(\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{f}}_{\mathbf{f}'} = \underbrace{(\mathbf{c}^\top \otimes \mathbf{I}) \cdot \mathbf{v}}_{u'}$$

More Efficient Construction: Rings

Main drawbacks of integer construction

More Efficient Construction: Rings

Main drawbacks of integer construction

- soundness amplification factor κ

More Efficient Construction: Rings

Main drawbacks of integer construction

- soundness amplification factor κ

Move to ring setting

More Efficient Construction: Rings

Main drawbacks of integer construction

- soundness amplification factor κ

Move to ring setting: challenge space = set of short polynomials

More Efficient Construction: Rings

Main drawbacks of integer construction

- soundness amplification factor κ

Move to ring setting: challenge space = set of short polynomials

$$\text{exponential size} \implies \kappa = 1$$

More Efficient Construction: Rings

Main drawbacks of integer construction

- soundness amplification factor κ

Move to ring setting: challenge space = set of short polynomials

$$\text{exponential size} \implies \kappa = 1$$

For concrete efficiency

$$r = \sqrt[3]{d} \quad + \quad \text{techniques from previous works}$$

Post-Quantum Security

Knowledge extraction via rewinding.

Post-Quantum Security

Knowledge extraction via rewinding. ⚠

Post-Quantum Security

Knowledge extraction via rewinding. ⚠

Advances in recent works [CMSZ22; LMS22]

Post-Quantum Security

Knowledge extraction via rewinding. ⚠

Advances in recent works [CMSZ22; LMS22]

uniformly sampled vs *correlated* challenges

Post-Quantum Security

Knowledge extraction via rewinding. ⚠

Advances in recent works [CMSZ22; LMS22]

uniformly sampled vs *correlated* challenges

Show how to bypass such hurdles using techniques from [BBK22].

Concretely Efficient Lattice-based Polynomial Commitment from Standard Assumptions

Intak Hwang, **Jinyeong Seo**, Yongsoo Song

Seoul National University

Objective of Our PCS

Large coefficient modulus

- Some lattice primitives (e.g., HE) use polynomial rings with large moduli.
- PCS that can handle these cases is needed.

Zero-knowledge w/o rejection

- Rejection sampling method [Lyu12] is unsuitable for some cases (e.g., MPC).
- Hint-MLWE method [KLS+23] is a promising alternative.

Notations

- $R := \mathbb{Z}[X]/(X^d + 1)$: Cyclotomic polynomial ring.
- q : Commitment modulus.
- p : Coefficient modulus.
- $D_{c+\Lambda,\sqrt{\Sigma}}$: Discrete Gaussian distribution over the coset $c + \Lambda$ with covariance matrix Σ.
- $\|\cdot\|_\infty$: Norm for polynomials (the largest coefficients).

Ajtai Commitment

- Our PCS is based on the Ajtai commitment.
- **Compressing:** Commitment length is shorter than message length.
- **Hiding:** Based on the MLWE problem.
- **Binding:** Based on the MSIS problem.

$$\boldsymbol{c} = \mathbf{A}_0 \boldsymbol{m} + \mathbf{A}_1 \boldsymbol{r} \pmod{q}$$

- $\boldsymbol{m} \in R^{\ell}$: Message, $\boldsymbol{r} \in R^{\nu}$: Randomness, $\boldsymbol{c} \in R_q^{\mu}$: Commitment
- $\mathbf{A}_0 \in R_q^{\mu \times \ell}, \mathbf{A}_1 \in R_q^{\mu \times \nu}$: CRS matrices, μ: MSIS rank, ν: MLWE rank
- Binding holds when $(\boldsymbol{m}, \boldsymbol{r})$ have small norms due to the MSIS problem.

Encoding for Large Coefficient Modulus

Issues

- The coefficient modulus p is too large to be committed directly.
- An encoding method is needed that maps large coefficients to small messages.
- The encoding must be a homomorphism for polynomial evaluations.

Encoding for Large Coefficient Modulus

Our solution

- We employ the following encoding map from $\mathbb{Z}_p^k$ to $R/(X^k - b)R = R_{X^k-b}$ when $p = b^{d/k} + 1$.

$$\mathtt{Ecd} : (a_0, \ldots, a_{k-1}) \mapsto \sum_{i=0}^{k-1} \left(\sum_{j=0}^{d/k-1} a_{i,j} X^{jk} \right) X^i$$

- Here, $a_i = \sum_{j=0}^{d/k-1} a_{i,j} b^j$ is the base-b representation of $a_i \in \mathbb{Z}_p$, so the norm of output polynomial is <u>bounded by b</u>.
- $\mathtt{Ecd}$ is an <u>isomorphism</u> since $\mathbb{Z}_p^k \cong R/(X^k - b)R$.
- b can be set much smaller than p (e.g., $b \approx 2^{16} \ll p \approx 2^{256}$).
- q is determined by the value of b, rather than p. (e.g., $q \approx 2^{112}$)

Encoding for Large Coefficient Modulus

More Details

- Using the encoding map $\mathtt{Ecd}$, a vector $\vec{a} = (\vec{a}_0, \dots, \vec{a}_{\ell-1}) \in (\mathbb{Z}_p^k)^\ell$ can be committed as follows:

$$\mathbf{Com}(\vec{a}) = \mathbf{A}_0 \begin{bmatrix} \mathtt{Ecd}(\vec{a}_0) \\ \vdots \\ \mathtt{Ecd}(\vec{a}_{\ell-1}) \end{bmatrix} + \mathbf{A}_1 \boldsymbol{r}$$

- It supports linear homomorphism for $\alpha \in \mathbb{Z}_p$ where $\mathtt{Ecd}(\alpha) = \mathtt{Ecd}(\alpha, \dots, \alpha)$:

$$\mathtt{Com}(\vec{a} + \alpha \cdot \vec{b}) = \mathtt{Com}(\vec{a}) + \mathtt{Ecd}(\alpha) \cdot \mathtt{Com}(\vec{b})$$

since $\mathtt{Ecd}(\vec{a}_i + \alpha \cdot \vec{b}_i) = \mathtt{Ecd}(\vec{a}_i) + \mathtt{Ecd}(\alpha) \cdot \mathtt{Ecd}(\vec{b}_i) \pmod{X^k - b}$.

Proof of Knowledge (PoK)

- PoK is required for the knowledge-soundness of PCS.
- PoK for Ajtai commitment is instantiated using a 3-move sigma protocol.

Prover$(\boldsymbol{m}, \boldsymbol{r})$ Verifer$(\mathbf{A}_0\boldsymbol{m} + \mathbf{A}_1\boldsymbol{r})$

$$\xrightarrow{\quad \mathbf{A}_0\boldsymbol{u} + \mathbf{A}_1\boldsymbol{y} \quad}$$

$$\xleftarrow{\quad c \quad}$$

$$\xrightarrow{\quad \boldsymbol{v} = \boldsymbol{u} + c \cdot \boldsymbol{m}, \quad \boldsymbol{z} = \boldsymbol{y} + c \cdot \boldsymbol{r} \quad}$$

- To achieve zero-knowledge, $(\boldsymbol{v}, \boldsymbol{z})$ should be simulatable.

Hint-MLWE

Definition

The Hint-MLWE problem asks an adversary $\mathcal{A}$ to distinguish between the following two distributions, where $\mathbf{A} \leftarrow \mathcal{U}(R_q^{\ell \times \nu})$, $\vec{\boldsymbol{u}} \leftarrow \mathcal{U}(R_q^{\ell})$, $\vec{\boldsymbol{r}} \leftarrow \chi$, $\vec{\boldsymbol{y}}_i \leftarrow \psi$, and $\vec{\boldsymbol{z}}_i = \boldsymbol{c}_i \cdot \vec{\boldsymbol{r}} + \vec{\boldsymbol{y}}_i$ for $0 \leq i < n$:

- $\left(\mathbf{A}, [\mathbf{A} \mid \mathbf{I}_\ell]\vec{\boldsymbol{r}}, \underline{\vec{\boldsymbol{z}}_0, \dots, \vec{\boldsymbol{z}}_{n-1}}\right)$
- $\left(\mathbf{A}, \vec{\boldsymbol{u}}, \underline{\vec{\boldsymbol{z}}_0, \dots, \vec{\boldsymbol{z}}_{n-1}}\right)$

- There is a reduction from the MLWE problem if χ and ψ are discrete Gaussian distributions $D^{\nu}_{\mathbb{Z}^d, \sigma \mathbf{I}}$ [KLS+23; MKM+22].
- The response $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{c} \cdot \boldsymbol{r}$ in PoK is simulatable using Hint-MLWE.

Randomized Encoding

Observation

- $\boldsymbol{v} = \boldsymbol{u} + c \cdot \boldsymbol{m}$ is not covered by Hint-MLWE.
- To apply a Hint-MLWE-like approach, $\boldsymbol{m}$ needs to be a random variable drawn from a discrete Gaussian distribution.
- The correctness of PCS is maintained if $\boldsymbol{m}$ is replaced by $\boldsymbol{m}'$, where $\boldsymbol{m} = \boldsymbol{m}' \pmod{X^k - b}$.
- The set of such $\boldsymbol{m}'$ forms a coset of a lattice $\boldsymbol{m} + \Lambda$ (when interpreting a polynomial as a vector of coefficients).

Randomized Encoding

Our Solution

- Sample $\boldsymbol{m}' \leftarrow D_{\boldsymbol{m}+\Lambda,\sqrt{\Sigma}}$ and $\boldsymbol{u} \leftarrow D_{\mathbb{Z}^{d\ell},\sqrt{\Sigma}}$ so that they follow discrete Gaussian distributions.
- The commitment is given as $\mathbf{A}_0\boldsymbol{m}' + \mathbf{A}_1\boldsymbol{r}$ and the response $\boldsymbol{v}$ is given as $\boldsymbol{u} + c \cdot \boldsymbol{m}'$.
- By the convolution lemma [Pei10],

$$D_{\mathbb{Z}^{d\ell},\sqrt{\Sigma}} + c \cdot D_{\boldsymbol{m}+\Lambda,\sqrt{\Sigma}} \approx D_{\mathbb{Z}^{d\ell},\sqrt{(c+I)\Sigma}}$$

since $\boldsymbol{m} + \Lambda \subseteq \mathbb{Z}^{d\ell}$, so $\boldsymbol{v}$ is now simulatable.

Polynomial Evaluation

- Adapted from the square-root evaluation strategy for the Pedersen commitment [BCC+16].
- For a polynomial $f(X) = f_0 + f_1 X + \ldots f_{N-1} X^{N-1} \pmod{p}$,

$$f(x) = \begin{bmatrix} 1 & 1 & x^{\sqrt{N}} & \cdots & x^{N-\sqrt{N}} & x \end{bmatrix} \begin{bmatrix} 0 & g_1 & \cdots & g_{\sqrt{N}-1} \\ f_0 & f_1 & \cdots & f_{\sqrt{N}-1} \\ \vdots & \vdots & & \vdots \\ f_{N-\sqrt{N}} & f_{N-\sqrt{N}+1} & \cdots & f_{N-1} \\ -g_1 & -g_2 & \cdots & 0 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \vdots \\ x^{\sqrt{N}-1} \end{bmatrix}$$

- Each row vector is committed to as $\boldsymbol{c}_i$ $(0 \leq i < \sqrt{N}+2)$, and the evaluation proof is given by the opening of $\underline{\boldsymbol{c}_0 + \sum_{i=1}^{\sqrt{N}-1} \mathtt{Ecd}(x^{i\sqrt{N}}) \cdot \boldsymbol{c}_i + \mathtt{Ecd}(x) \cdot \boldsymbol{c}_{\sqrt{N}}}$.
- Proof size: $\widetilde{O}(\sqrt{N})$ Verification cost: $\widetilde{O}(\sqrt{N})$

Benchmark Results

Comparison with Brakedown [GLS+23] for $\log p \approx 255$

N		2^{19}	2^{21}	2^{23}	2^{25}
Prover(seconds)	**Ours**	0.97	3.47	13.0	50.9
	Brakedown	0.60	2.41	9.85	39.2
Verifier(seconds)	**Ours**	0.14	0.27	0.53	1.07
	Brakedown	0.15	0.30	0.61	0.70
Communication(MB)	**Ours**	6.07	11.9	23.6	47.5
	Brakedown	10.0	15.8	27.1	49.2

Benchmark Results

Comparison with SLAP [AFL+24]

	N	$\log p$	Proof size
Ours	2^{20}	255	**8.93** MB
SLAP	2^{20}	276	36.5 MB

Thank you!

eprint: `https://eprint.iacr.org/2024/306`

github: `https://github.com/SNUCP/celpc`

Greyhound: Fast Polynomial Commitments from Lattices

Ngoc Khanh Nguyen and **Gregor Seiler**

Aug 19, 2024

King's College London and IBM Research Europe

Third Polynomial Commitment

Evaluation of $f(X) = f_0 + f_1 X + f_2 X^2 + f_3 X^3$:

- Write evaluation as quadratic form

$$f(\alpha) = \begin{pmatrix} 1 & \alpha \end{pmatrix} \begin{pmatrix} f_0 & f_2 \\ f_1 & f_3 \end{pmatrix} \begin{pmatrix} 1 \\ \alpha^2 \end{pmatrix}$$

- Send $\begin{pmatrix} w_0 & w_1 \end{pmatrix} = \begin{pmatrix} 1 & \alpha^2 \end{pmatrix} \begin{pmatrix} f_0 & f_2 \\ f_1 & f_3 \end{pmatrix}$
- Randomly linear-combine columns $\begin{pmatrix} f_0 \\ f_1 \end{pmatrix} + c \begin{pmatrix} f_2 \\ f_3 \end{pmatrix}$
- Use Labrador to prove $w_0 + c w_1 = \begin{pmatrix} 1 & \alpha^2 \end{pmatrix} \left(\begin{pmatrix} f_0 \\ f_1 \end{pmatrix} + c \begin{pmatrix} f_2 \\ f_3 \end{pmatrix} \right)$

  and $f(\alpha) = \begin{pmatrix} w_0 & w_1 \end{pmatrix} \begin{pmatrix} 1 \\ \alpha^3 \end{pmatrix}$

Differences to the other schemes

- We don't recurse and immediately use Labrador — square root is good enough
- We hide the large cost of $\vec{w}$ by committing to it and having it part of the Labrador witness
- We use an optimized parameterization to minimize proof size

Implementation

We provide a fully vectorized implementation for AVX-512 in C with intrinsics — finally online: `github.com/lattice-dogs/labrador`

The polynomial operations in lattice-based cryptography profit massively from vectorization, which is not really accessible from plain C

So restricting to plain C implementations would give a distorted picture of real-world performance

Important building blocks:

- Polynomial arithmetic
- Johnson-Lindenstrauss projection
- Parameter finding

Polynomial arithmetic

For small proof sizes we need NTT-unfriendly q, i.e. $q \equiv 5 \pmod 8$

Still: Fastest arithmetic by using NTT-based approach via modulus switching:

1. Lift polynomials to $\mathbb{Z}[X]/(X^{64}+1)$
2. Operate in $\mathbb{Z}_{p_i}[X]/(X^{64}+1)$ for many small NTT-friendly p_i, using NTT
3. Apply explicit CRT to obtain centered result mod $P = \prod_i p_i$
4. Reduce mod q — correct if operation on lifted polynomials would result in coefficients bounded by $P/2$

This is usually faster even for a single multiplication

And results in large saving for high-dim matrix-vector products

Vector layout

For fast modular reduction mod p_i can use 16-bit or 52-bit multipliers on AVX-512

52-bit p_i don't give enough granularity so we opted for 16-bit arithmetic

To enable efficient transformation between multimodular representation and direct representation mod q we have implemented all mod q operations using 14-bit signed multiprecision arithmetic

So coefficient limbs align with the 16-bit coefficients mod p_i

Johnson-Lindenstrauss Projection

A crucial ingredient in lattice-based proofs are proofs of shortness $\|\vec{s}\| \leq \beta$

Have explored many approaches over the last years

Now pretty much settled for using random projections — both for l2-norm and infinity norm (binary)

Johnson-Lindenstrauss: Some linear projections tightly preserve l2-norm up to constants

Fast projection using Four Russians algorithm

$$\vec{p} = \begin{pmatrix} -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}$$

1. Precompute all 16 signed summations of s_0, s_1, s_2, s_3
2. For each row of matrix just look up correct summation

Four Russians on AVX-512

On AVX-512 can store the **16** summations in one vector register of **32** bit integers

Then simultaneously look up **16** summations at a time using a vector shuffle instruction

In lattice proofs we need not only multiply from the right for projection but also multiply from the left for randomly collapsing the matrix

The former only has to be performed by the prover whereas the later has to be performed by the prover and verifier, multiple times

So we optimize for the latter case

Results: Proof Sizes, Runtimes, Comparison

	2^{25}				2^{29}			
	size	comm	prove	verify	size	comm	prove	verify
Brakedown-PC	49′157 KB	36 s	3.21 s	0.703 s	181′948 KB	605 s	48.6 s	2.96 s
Ligero-PC	7′256 KB	83.9 s	3.13 s	0.338 s	28′631 KB	1590 s	51.6 s	1.57 s
FRI-PC	740 KB	168 s	185 s	0.041 s	—	—	—	—
CMNW	1′393 KB	—	—	—	3′983 KB	—	—	—
HSS	48′640 KB	30.9 s	19.6 s	1.07 s	198′656 KB	—	—	—
Greyhound	**47 KB**	**1.84 s**	**0.788 s**	**0.239 s**	**52 KB**	**72 s**	**21.7 s**	**1.61 s**

Thank you!

`github.com/lattice-dogs/labrador`