# Cyclo: Lightweight Lattice-based Folding via Partial Range Checks

Albert Garreta, Helger Lipmaa, **Urmas Luhaäär**, Michał Osadnik

April 2026

# Reductions of knowledge

A reduction of knowledge [KP23] is a protocol that reduces the knowledge of a witness of some instance of a relation to knowledge of a witness to some instance of some other relation.

Let $P$ denote the prover and $V$ the verifier and let let $G$ generate the parameters.

**Definition**

*Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be relations. Let $\Pi$ be a protocol where the prover gets input $(x, w)$ and the verifier gets $x$. Let the prover's output be $(x', w')$ and the verifier's is $x'$.*

*Then $\Pi$ is a reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_2$ if it satisfies completeness, knowledge soundness and public reducibility defined below.*

# Reductions of knowledge

1. Completeness: Given $(x, w) \in \mathcal{R}_1$, if both $P$ and $V$ follow the protocol honestly then the verifier's output instance is equal to the prover's output instance and if the prover's output is $(x', w')$, then $(x', w') \in \mathcal{R}_2$.
2. Knowledge-soundness: The protocol $\Pi$ is *knowledge sound* with knowledge error $\kappa$ if there exists an expected polynomial-time extractor $E$ such that for all instances $x$ of $\mathcal{R}_1$ and PPT $P^*$ which together with $V$ outputs $(y, u) \in \mathcal{R}_2$ with probability $\varepsilon > \kappa$, the extractor outputs an element $w$ such that $(x, w) \in \mathcal{R}_1$ with probability at least $\varepsilon - \kappa$.
3. Public reducibility: There exists a deterministic polynomial-time function $\phi$ such that for any expected polynomial time $P^*$, given the public parameters pp, input statement $x$, $(y, u) \leftarrow \langle P^*, V \rangle$ and the transcript of the interaction $\langle P^*, V \rangle$, $\phi(\mathsf{pp}, x, \mathsf{tr}) = y$.

# Examples of RoK

**Definition**

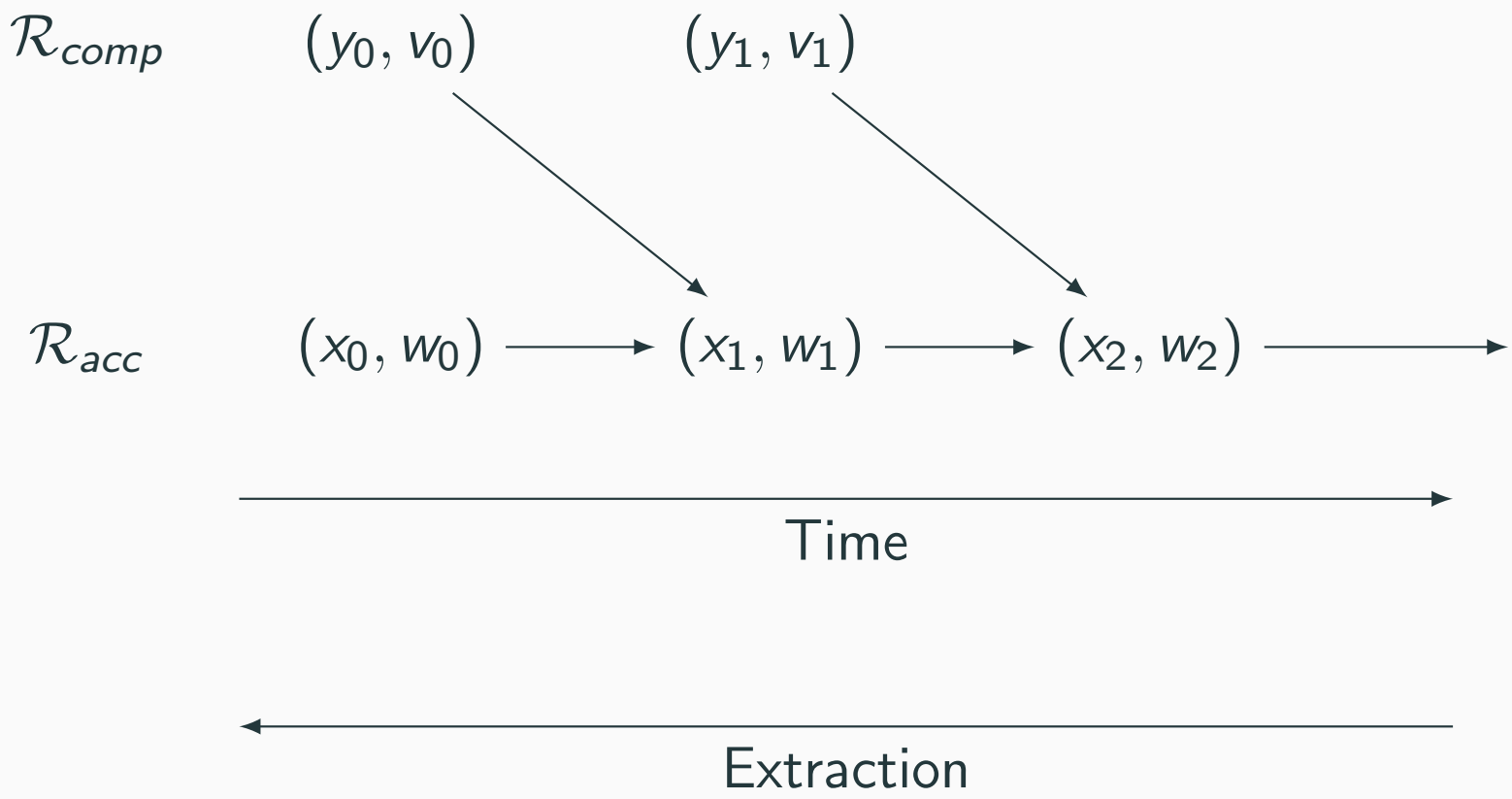
*Let $\mathcal{R}$ be a relation and let $1$ denote the trivial relation where every element is a witness to every instance. Then a proof of knowledge is a reduction of knowledge from $\mathcal{R}$ to $1$.*

**Definition**

*Let $\mathcal{R}_{acc}$ and $\mathcal{R}_{comp}$ be relations. Then a folding scheme is a reduction of knowledge from $\mathcal{R}_{acc} \times \mathcal{R}_{comp}$ to $\mathcal{R}_{acc}$.*

If in the above definition, there are multiple instances of $\mathcal{R}_{acc}$ or $\mathcal{R}_{comp}$, then we call the protocol a multi-folding scheme.

# Folding schemes



# Ajtai commitments

**Definition**

*Let $R_q = \mathbb{F}_q[X]/(X^d + 1)$ and $m, n \in \mathbb{N}$. Then the Ajtai commitment scheme has the key generation algorithm*

$$A \leftarrow \mathsf{keyGen}(m, n, B),$$

*where $A$ is a uniformly generated element of $\mathrm{Mat}_{n,m}(R_q)$ and the message space is $\{f \in R_q^m \mid \|f\|_\infty < B\}$.*

*The commitment algorithm is*

$$\mathsf{comm}(A, f) = Af.$$

The Ajtai commitment scheme is binding under the assumption that $\mathrm{MSIS}_{q,d,m,n,2B}$ is hard.

# Linear relations

$$\Xi^{\mathsf{lin}}_{\mathsf{A},(\mathsf{M}_i)_{i\in[k]},a,n,m,B} := \left\{ \begin{array}{l} \underline{((\mathsf{r}_i)_{i\in[k]},(\mathsf{b}_i)_{i\in[n]},\mathsf{y}),\mathsf{w}:} \\ \left(\mathsf{M}_i \in R_q^{m_i\times m},\ \mathsf{r}_i \in R_{q^e}^{\log m_i}\right)_{i\in[k]},\ (\mathsf{b}_i \in R_{q^e}^{\log m})_{i\in[n]}, \\ \mathsf{A} \in R_q^{a\times m},\ \mathsf{y} = \begin{pmatrix} \overline{\mathsf{y}} \in R_q^a \\ \underline{\mathsf{y}} \in R_{q^e}^{k+n} \end{pmatrix},\ \mathsf{w} \in R_q^m,\ \|w\| \le B \\ \begin{pmatrix} \mathsf{A} \\ \mathtt{tensor}(\mathsf{r}_0)^{\mathsf{T}}\mathsf{M}_0 \\ \vdots \\ \mathtt{tensor}(\mathsf{r}_{k-1})^{\mathsf{T}}\mathsf{M}_{k-1} \\ \mathtt{tensor}(\mathsf{b}_0)^{\mathsf{T}} \\ \vdots \\ \mathtt{tensor}(\mathsf{b}_{n-1})^{\mathsf{T}} \end{pmatrix} \mathsf{w} = \mathsf{y} \bmod q \end{array} \right\}$$

# Lattice based folding

- Modern lattice-based folding schemes (e.g. [BC26; BC25; NS25]) use random linear combinations of Ajtai commitments.
- The problem is that this causes the witness-norm to grow. This poses a problem in both the correctness direction and extraction direction.
- If we fold two statements and the norm becomes too big, we might not get an instance-witness pair of the target relation i.e. we lose correctness.
- If our witness is too big, Ajtai commitments can lose their binding i.e. we lose extractability.

# Lattice-based folding

- To solve norm-growth in the correctness direction, a decomposition step is applied.
- To solve extractability, a shortness proof is applied to the witness.

$$\left.\begin{array}{l} \mathcal{R}_{acc} \xrightarrow{\text{norm-check}} \mathcal{R}'_{acc} \\ \mathcal{R}_{comp} \xrightarrow{\text{norm-check}} \mathcal{R}'_{comp} \end{array}\right\} \xrightarrow{\text{fold}} \mathcal{R}_{folded} \xrightarrow{\text{decomposition}} \mathcal{R}_{acc}$$

# Cyclo

- Cyclo ignores norm-checks and decomposition on the accumulation relation.
- If the computation instance is preprocessed then we can also ignore the decomposition step on the computation relation.

$$\left.\begin{array}{l} \mathcal{R}_{acc} \\ \mathcal{R}''_{comp} \end{array}\right\} \xrightarrow{\text{fold}} \mathcal{R}_{acc}, \quad \mathcal{R}_{comp} \xrightarrow{\text{decomposition}} \mathcal{R}'_{comp} \xrightarrow{\text{norm-check}}$$

or

$$\left.\begin{array}{l} \mathcal{R}_{acc} \\ \mathcal{R}''_{comp} \end{array}\right\} \xrightarrow{\text{fold}} \mathcal{R}_{acc}, \quad \mathcal{R}_{comp} \xrightarrow{\text{norm-check}}$$

# Amortized norm-refreshing folding scheme

- The drawback of Cyclo is that there is a linear norm growth in both extraction and correction.
- One solution is to form time-to-time refresh the norm with a different lattice-based folding scheme such as Latticefold+.

$$\left.\begin{matrix}\mathcal{R}_{comp}\\ \mathcal{R}_{acc}\end{matrix}\right\} \xrightarrow{\text{Cyclo}} \left.\begin{matrix}\mathcal{R}_{comp}\\ \mathcal{R}_{acc}\end{matrix}\right\} \xrightarrow{\text{Cyclo}} \cdots \xrightarrow{\text{LF+}} \left.\begin{matrix}\mathcal{R}_{comp}\\ \mathcal{R}_{acc}\end{matrix}\right\} \xrightarrow{\text{Cyclo}} \cdots$$

# Range check

**Range Test**

$\Pi_b^{\mathsf{range}} : \big(((\mathbf{r}_i)_{i\in[k]}, (\mathbf{b}_i)_{i\in[n]}, \mathbf{y}), \mathbf{w}\big) \in \Xi_{n,b}^{\mathsf{lin}} \to \Xi_{n+1,b}^{\mathsf{lin}}$

1. Prover defines $f := \mathsf{MLE}[\mathtt{cf}(\mathbf{w})] \in \mathbb{Z}_q[X_0, \ldots, X_{\ell-1}]$ with $\ell = \lceil \log(m\varphi) \rceil$. Let $\mathtt{tensor}(\cdot) \in \mathbb{F}_{q^e}^{m\varphi}$ denote the multilinear evaluation tensor such that $\langle \mathtt{tensor}(\mathbf{z}), \mathtt{cf}(\mathbf{w}) \rangle = \mathsf{MLE}[\mathtt{cf}(\mathbf{w})](\mathbf{z})$ for all $\mathbf{z} \in \mathbb{F}_{q^e}^{\ell}$. We view $\mathtt{tensor} : \mathbb{F}_{q^e}^{\ell} \to \mathbb{F}_{q^e}^{m\varphi}$ as the standard Boolean-ML extension tensor.
2. The verifier samples $\boldsymbol{\eta} = (\eta_0, \eta_1, \ldots, \eta_{\ell-1}) \leftarrow \mathbb{F}_{q^e}$ and sets $\omega(X) := \mathsf{eq}(X; \boldsymbol{\eta})$.
3. Prover sets $\hat{f}(X) := \prod_{j=-b}^{b} (f(X) - j) \cdot \omega(X) \in \mathbb{F}_{q^e}[X_0, \ldots, X_{\ell-1}]$.
4. The parties run sum-check over $\mathbb{F}_{q^e}$, reducing $\sum_{\mathbf{z}\in\{0,1\}^\ell} \hat{f}(\mathbf{z}) \stackrel{?}{=} 0$ to $\hat{f}(\mathbf{u}) \stackrel{?}{=} s$, where $\mathbf{u}$ is the challenge vector sampled by the verifier and $s \in \mathbb{F}_{q^e}$ is the claimed leaf value sent by the prover.
5. Prover sets $\tilde{t} := \langle \mathbf{u}', \mathbf{w} \rangle \in \mathcal{R}_{q^e}$ for $\mathbf{u}' := \mathtt{cf}_\vee^{-1}(\mathtt{tensor}(\mathbf{u}))$ and sends it to the verifier (here $\mathtt{cf}_\vee^{-1}(\mathtt{tensor}(\mathbf{u})) \in \mathcal{R}_{q^e}^{m}$ so the inner product lies in $\mathcal{R}_{q^e}$).
6. Verifier computes $t := \mathsf{Trace}(\tilde{t}) \in \mathbb{F}_{q^e}$ and checks $s \stackrel{?}{=} \omega(\mathbf{u}) \cdot \prod_{j=-b}^{b} (t - j)$. By Lemma 1 (dual-basis trace), $\mathsf{Trace}(\langle \mathtt{cf}_\vee^{-1}(\mathtt{tensor}(\mathbf{u})), \mathbf{w} \rangle) = \mathsf{MLE}[\mathtt{cf}(\mathbf{w})](\mathbf{u})$. Since $\mathsf{Trace}(\tilde{t}) = \mathsf{MLE}[\mathtt{cf}(\mathbf{w})](\mathbf{u}) = f(\mathbf{u})$, the leaf check equals $\hat{f}(\mathbf{u}) \stackrel{?}{=} s$.
7. Prover and verifier output: $\left(\left((\mathbf{r}_i)_{i\in[k]}, ((\mathbf{b}_i)_{i\in[n]}, \mathbf{u}'), \begin{pmatrix} \mathbf{y} \\ \tilde{t} \end{pmatrix}\right), \mathbf{w}\right) \in \Xi_{n+1,b}^{\mathsf{lin}}$.

# Extension commitment

**Extension commitment**

$\Pi^{\mathsf{ext}}_{b,\mathcal{C}} : \left(((\mathbf{r}_i)_{i\in[k]}, (\mathbf{b}_i)_{i\in[n]}, \mathbf{y}), \mathbf{w}\right) \in \Xi^{\mathsf{lin}}_{\mathbf{A},(\mathbf{M}_i)_{i\in[k]},a,n,m,B} \to \Xi^{\mathsf{lin}}_{\mathbf{R},(\widehat{\mathbf{M}}_i)_{i\in[k+1]},a',n,m\ell,b}$

1. The prover splits the witness: $\mathbf{v}^\intercal = (\mathbf{w}_0^\intercal, \ldots, \mathbf{w}_{\ell-1}^\intercal)$, where $\mathbf{w}_i$ are such that $\|\mathbf{w}_i\|_\infty \leq b$ for all $i \in [\ell]$ and $\mathbf{w} = \sum_{i=0}^{\ell-1} \mathbf{w}_i b^i$.
2. The prover sends the extended commitment $\mathbf{t} := \mathbf{R}\mathbf{v} \in \mathcal{R}_q^{a'}$ to the verifier.
3. The verifier samples $\widehat{\mathbf{c}} \leftarrow \mathcal{C}^{\ell_C}$ and sets $\mathbf{c} := \mathtt{tensor}(\widehat{\mathbf{c}}) \in \mathcal{R}_{q^e}^{a}$.
4. Let
   (i) $\widehat{\mathbf{b}}_i^\intercal := \left(((2b)^0, (2b)^1, \ldots, (2b)^{\ell-1}) \otimes \mathbf{b}_i^\intercal\right) \in \mathcal{R}_{q^e}^{\lceil \log m \rceil + \ell} \quad \forall\, i \in [k]$,
   (ii) $\widehat{\mathbf{M}}_i := ((2b)^0, (2b)^1, \ldots, (2b)^{\ell-1}) \otimes \mathbf{M}_i \in \mathcal{R}_q^{m_i \times m\ell} \quad \forall\, i \in [k]$,
   (iii) $\widehat{\mathbf{M}}_k := ((2b)^0, (2b)^1, \ldots, (2b)^{\ell-1}) \otimes \mathbf{A} \in \mathcal{R}_q^{a \times m\ell}$,
   The prover and verifier output the augmented instance

$$\left(((\mathbf{r}_i)_{i\in[k]}, \mathbf{c}), (\widehat{\mathbf{b}}_i)_{i\in[n]}, \widehat{\mathbf{y}}), \mathbf{v}\right) \in \Xi^{\mathsf{lin}}_{\mathbf{R},(\widehat{\mathbf{M}}_i)_{i\in[k+1]},a',n,m\ell,b}$$

   for $\widehat{\mathbf{y}}^\intercal := \left(\mathbf{t}^\intercal, y_a, \ldots, y_{a+k-1}, \langle \mathbf{c}, (y_0, \ldots, y_{a-1})\rangle, y_{a+k}, \ldots, y_{a+k+n-1}\right)$.

# Full protocol

**Folding Scheme**

$$\Pi^{\mathsf{fs}}_{b,\mathcal{D},\mathcal{C}} : \left( \begin{array}{c} \big(((\mathbf{r}_i)_{i\in[k]}, \mathbf{b}, \mathbf{y}), \mathbf{v}\big) \in \Xi^{\mathsf{lin}}_{\mathbf{R},(\widehat{\mathbf{M}}_i)_{i\in[k+1]},a',1,m\log_{2b}2B,\beta'} \\ \big(((\mathbf{r}'_{i,j})_{i\in[k]}, (\mathbf{b}'_{i,j})_{i\in[n]}, \mathbf{y}'_j), \mathbf{w}'_j\big)_{j\in[L]} \in (\Xi^{\mathsf{lin}}_{\mathbf{A},(\mathbf{M}_i)_{i\in[k]},a',n,m,B})^L \end{array} \right) \to \Xi^{\mathsf{lin}}_{\mathbf{R},(\widehat{\mathbf{M}}_i)_{i\in[k+1]},a',1,m\log_{2b}2B,\beta+Lb\gamma}$$

1. Prover and verifier compute for all $j \in [L]$: // see Fig. 2

$$\big(((\mathbf{r}''_{i,j})_{i\in[k+1]}, (\mathbf{b}'_{i,j})_{i\in[n]}, \mathbf{t}'_j), \mathbf{v}'_j\big) \leftarrow \Pi^{\mathsf{ext}}_{b,\mathcal{C}}\big(\big(((\mathbf{r}'_{i,j})_{i\in[k]}, (\mathbf{b}'_{i,j})_{i\in[n]}, \mathbf{y}'_j), \mathbf{w}'_j\big)\big).$$

2. Prover and verifier compute for all $j \in [L]$: // see Fig. 1

$$\big(((\mathbf{r}''_{i,j})_{i\in[k+1]},\ (\mathbf{b}''_{i,j})_{i\in[n+1]},\ \mathbf{t}'_j), \mathbf{v}'_j\big) \leftarrow \Pi^{\mathsf{range}}_b\left(\big(((\mathbf{r}''_{i,j})_{i\in[k+1]},\ (\mathbf{b}'_{i,j})_{i\in[n]}, \mathbf{t}'_j), \mathbf{v}'_j\big)\right).$$

# Full protocol

3. Verifier samples $\mathbf{d} \leftarrow \mathbb{F}_{q^e}^{\lceil \log \varphi(2+k+L(2+n+k)) \rceil}$.
4. Let $\tilde{m} := m \log_{2b} 2B$ and $\tilde{m}_i := m_i \log_{2b} 2B$ for $i \in [k+1]$. Prover and verifier express linear equalities concerning input relations as sum-check instances:
   (a) $\sum_{\mathbf{z} \in \{0,1\}^{\log \tilde{m}_i}} \mathsf{MLE}[\widehat{\mathbf{M}}_i \mathbf{v}'_j](\mathbf{z}) \mathsf{eq}(\mathbf{z}; \mathbf{r}''_{i,j}) = t'_{j,a'+i}, \quad i \in [k+1]\ j \in [L],$
   (b) $\sum_{\mathbf{z} \in \{0,1\}^{\log \tilde{m}_i}} \mathsf{MLE}[\widehat{\mathbf{M}}_i \mathbf{v}](\mathbf{z}) \mathsf{eq}(\mathbf{z}; \mathbf{r}_i) = y_{a'+i}, \quad i \in [k+1],$
   (c) $\sum_{\mathbf{z} \in \{0,1\}^{\log \tilde{m}}} \mathsf{MLE}[\mathbf{v}'_j](\mathbf{z}) \mathsf{eq}(\mathbf{z}; \mathbf{b}''_{i,j}) = t'_{j,a'+k+1+i}, \quad i \in [n+1]\ j \in [L],$
   (d) $\sum_{\mathbf{z} \in \{0,1\}^{\log \tilde{m}}} \mathsf{MLE}[\mathbf{v}](\mathbf{z}) \mathsf{eq}(\mathbf{z}; \mathbf{b}) = y_{a'+k+1},$

   Then, they batch the sum-check claims[a] with $\mathbf{d}$ and execute the sum-check protocol (over $\mathbb{F}_{q^e}$ NTT slots of $\mathcal{R}_{q^e}$) to reduce them to claims over a shared random point, i.e., $\left(((\hat{\mathbf{r}}_i)_{i \in [k]}, \hat{\mathbf{r}}, \tilde{\mathbf{y}}), \mathbf{v}\right) \in \Xi^{\mathsf{lin}}_{\mathbf{R}, (\widehat{\mathbf{M}}_i)_{i \in [k+1]}}$ and

$$\left(((\hat{\mathbf{r}}_i)_{i \in [k]}, \hat{\mathbf{r}}, \tilde{\mathbf{y}}'_j), \mathbf{v}'_j\right) \in \Xi^{\mathsf{lin}}_{\mathbf{A}, (\mathbf{M}_i)_{i \in [k]}} \text{ for } j \in [L].$$

   The new evaluation claims $\tilde{\mathbf{y}}[a', |\tilde{\mathbf{y}}|]$ and $\tilde{\mathbf{y}}'_j[a', |\tilde{\mathbf{y}}'_j|]$ (over $\mathcal{R}_{q^e}$) are sent to the verifier.
5. Verifier sends a short folding challenge $\mathbf{s} \leftarrow \mathcal{D}^L$.
6. Prover and verifier output:

$$\left(\left((\hat{\mathbf{r}}_i)_{i \in [k]}, \hat{\mathbf{r}}, \tilde{\mathbf{y}} + \sum_{j \in [L]} s_j \tilde{\mathbf{y}}'_j\right), \mathbf{v} + \sum_{j \in [L]} s_j \mathbf{v}'_j\right) \in \Xi^{\mathsf{lin}}_{\mathbf{R}, (\widehat{\mathbf{M}}_i)_{i \in [k+1]}, a', 1, \tilde{m}, \beta + Lb\gamma}$$

   where $\gamma$ is the expansion factor of the challenge set $\mathcal{D}$.

[a] Batching sum-check claims with different numbers of variables requires padding functions with fewer variables using unused auxiliary variables and later erasing the corresponding coordinates from the output evaluation point.

# From relations over fields to relations over rings

- In [NS25], an efficient way of embedding relations over fields to relations over rings was found.
- In our works we give a more streamlined and algebraic description of this result.
- Recall $R_q = \mathbb{F}_q[X]/\langle X^d + 1\rangle$ and fix some positive integer $k$.
- Then we have the $\mathbb{F}_q$-module homomorphism $\theta_k : R_q \to \mathbb{F}_q$ by $f(X) \mapsto f(k) \mod q$
- We can write any element $c \in \mathbb{F}_q$ uniquely as $c = c_0 + c_1 k + \cdots + c_n k^n$. Then writing $p_c(X) = c_0 + c_1 X + \cdots + c_n X^n$ gives us an element of $R_q$.
- Using these mappings, we can go between relations over $\mathbb{F}_q$ to ones over $R_q$.

*Thanks!*

# References


[KP23] Abhiram Kothapalli and Bryan Parno. **"Algebraic Reductions of Knowledge".** In: *Advances in Cryptology – CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part IV*. Santa Barbara, CA, USA: Springer-Verlag, 2023, pp. 669–701. ISBN: 978-3-031-38550-6. DOI: 10.1007/978-3-031-38551-3_21. URL: https://doi.org/10.1007/978-3-031-38551-3_21.

[BC25] Dan Boneh and Binyi Chen. **"LatticeFold+: Faster, Simpler, Shorter Lattice-Based Folding for Succinct Proof Systems".** In: *Advances in Cryptology – CRYPTO 2025*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Cham: Springer Nature Switzerland, 2025, pp. 327–361. ISBN: 978-3-032-01907-3.