latticefold

Latticefold: How to reduce witness size under Ajtai commitment

Recall that in Hypernova's folding, we can fold a committed CCS instance and a linear committed CCS instance to one LCCCS instance using an additively homomorphic commitment.

To construct a folding scheme from lattice, the difficulty is ensuring that the committed witnesses are low norm through many rounds of folding.
As we have had the framework of Hyprenova(shown below), this time we will focus on this challenge is solved in Latticefold.

1. Committed CCS to Linear CCCS
2. Fold: LCCCS x LCCCS to LCCCS

1. Ajtai commitment

Let $\mathcal{R} = \mathbb{Z}/(X^d + 1)$ and $\mathcal{R}_q = \mathbb{Z}_q/(X^d + 1)$.

1. For a message vector $\mathbf{x} \in \mathcal{R}^m$ with infinity norm bound $|\mathbf{x}|_\infty \leq B$, the **Ajtai commitment** process is as follows:
   - **KeyGen**: Given the security parameter λ, generate a commitment key $\mathbf{A} \in \mathcal{R}_q^{n \times m}$.
   - **Com**: Given the commitment key $\mathbf{A} \in \mathcal{R}_q^{n \times x}$ and a message $\mathbf{x} \in \mathcal{R}^m$, compute the commitment $cm = \mathbf{A}\mathbf{x}$.
2. Ajtai commitment is additively homomorphic.
   - $Com(\mathbf{x}_1) + Com(\mathbf{x}_2) = \mathbf{A}(\mathbf{x}_1 + \mathbf{x}_2)$.
   - the norm grows: $|\mathbf{x}_1 + \mathbf{x}_2|_\infty \leq 2B$

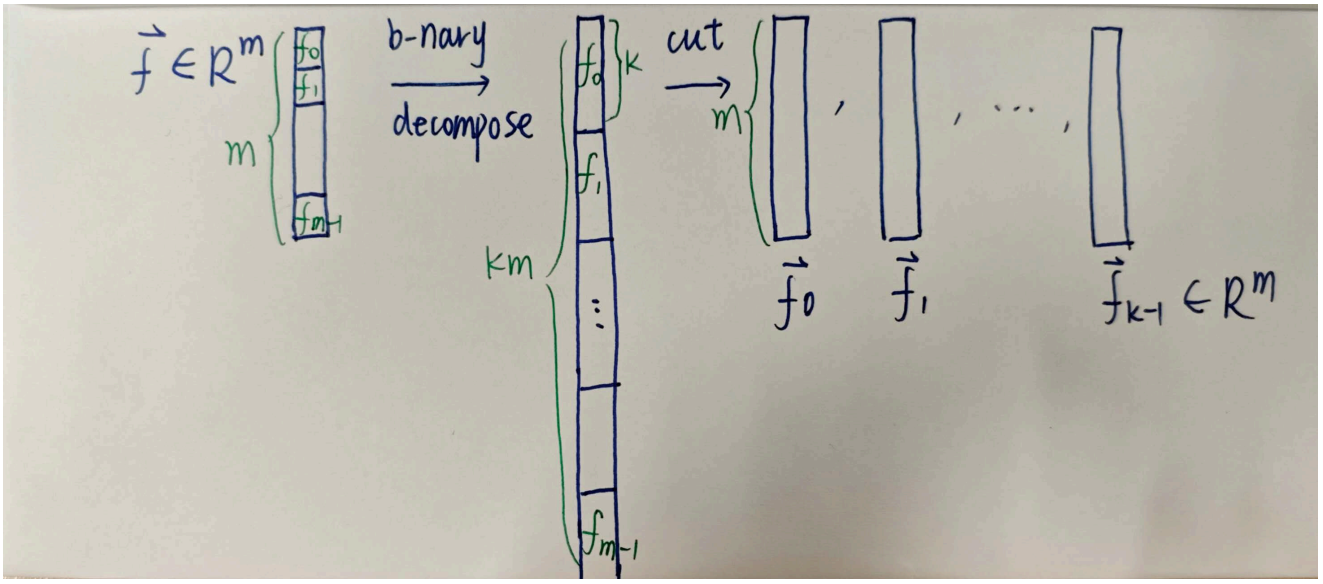
2. Control the Norm

We discuss how to control the witness size while folding.
It consists of 2 steps: Decomposition and Fold.

Decomposition

For a witness $\vec{f} \in \mathcal{R}^m$ of bounded norm B, decompose it into a tuple of vectors $\vec{f}_0, \ldots, \vec{f}_{k-1} \in \mathcal{R}^m$ of lower norm $b := \lceil B^{1/k} \rceil$.

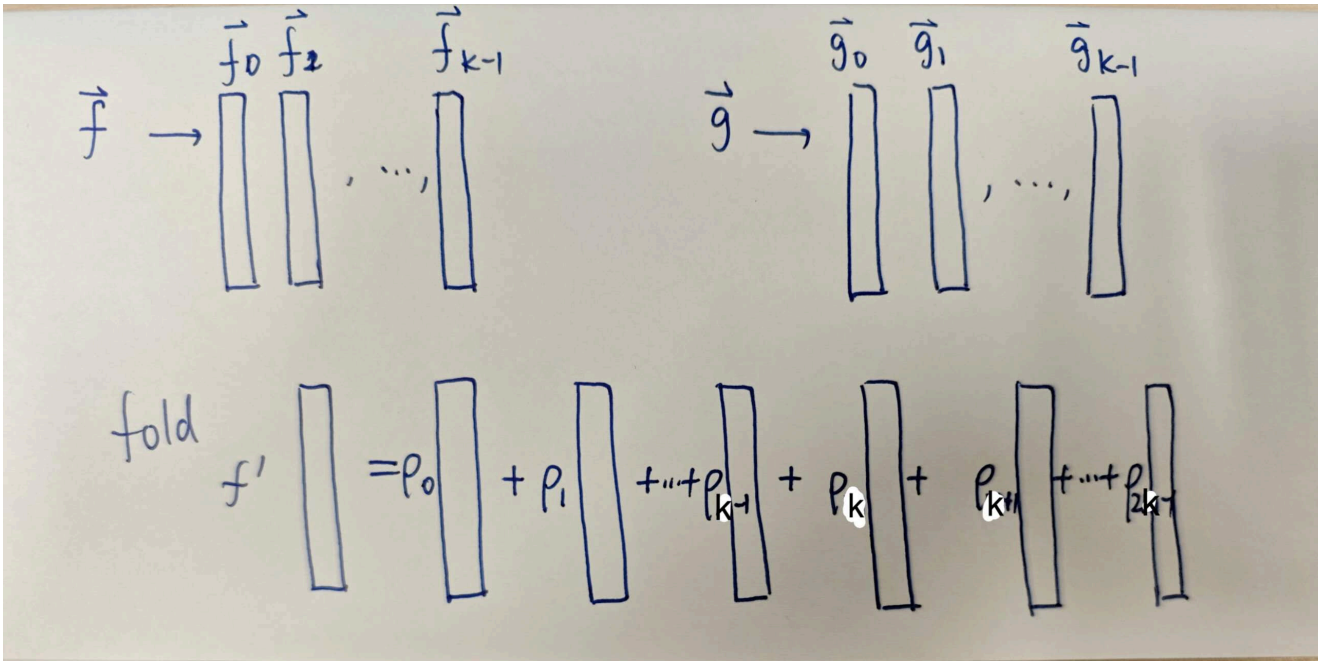
This decomposition works by writing every entry of $\vec{f}$ in base b, so that the original $\vec{f}$ satisfies $\vec{f} = \vec{f}_0 + b \cdot \vec{f}_1 + \ldots + b^{k-1} \cdot \vec{f}_{k-1}$, and each of the k vectors has norm less than b.



Fold the witness

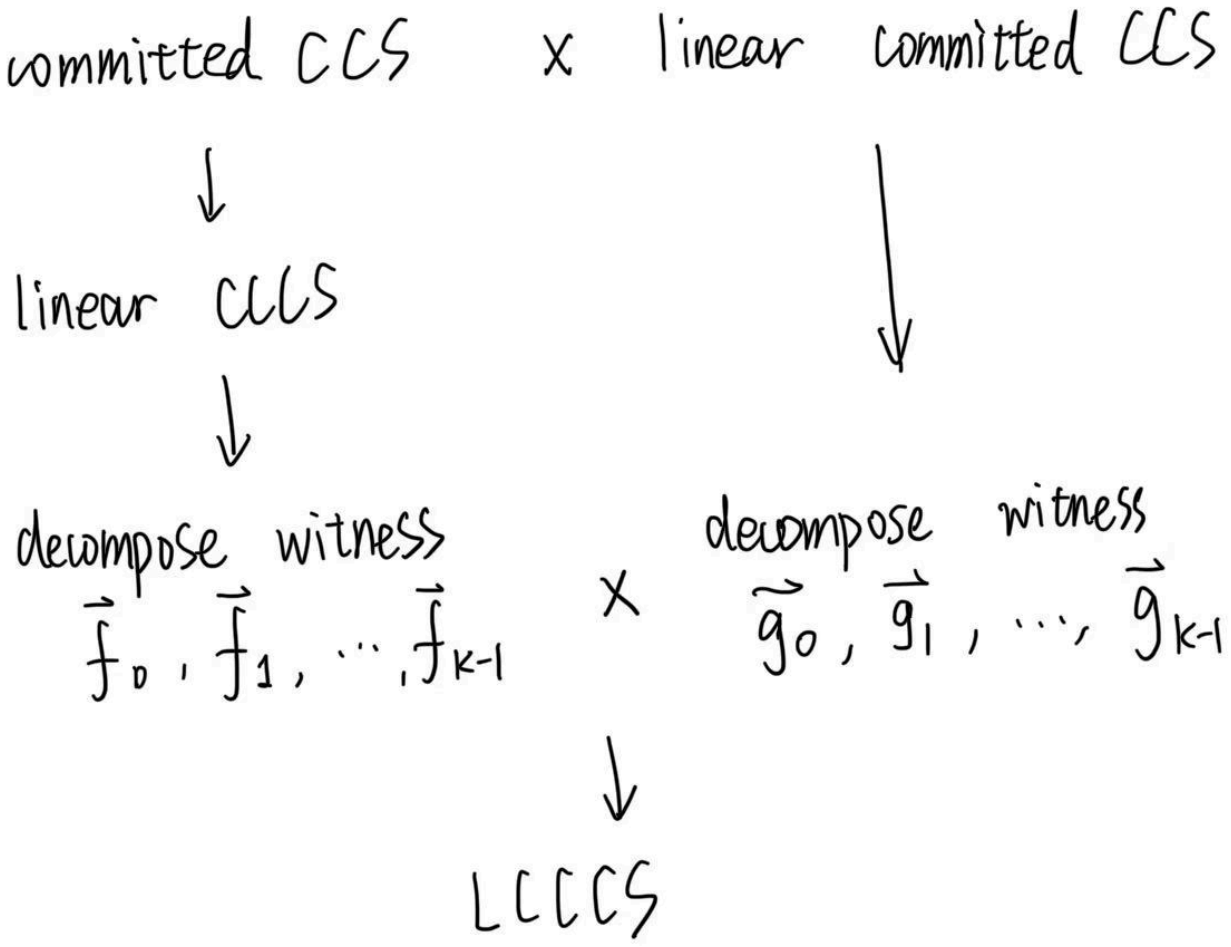
Given 2 decomposed witnesses $(\vec{f}_0, \ldots, \vec{f}_{k-1})$ and $(\vec{g}_0, \ldots, \vec{g}_{k-1})$ with lower bounded norm b.

We compute a random linear combination of these $2k$ vectors using a random vector of weights $\vec{\rho}$ sampled as $\vec{\rho} \xleftarrow{\$} \mathcal{C}^{2k}_{small}$.



By carefully selecting the low norm of ρ, we can make sure that the final folded witness $\vec{f'} := \sum_{i=1}^{2k} \rho_i \vec{f}_i$ has norm at most B.

3. Latticefold



1. Committed CCS to Linear CCCS
   - use Ajtai commitment
   - fix CCCS at $\mathbf{0}$ to get linear CCCS
2. Decompose LCCCS witness
   - b-nary decomposition into k witnesses
   - the verifier check decomposition
3. Fold LCCCS x LCCCS to LCCCS
   - fold $2k$ LCCCS (with norm bound b) into one LCCCS (with norm bound B)
   - the verifier check
     - 2 LCCCS instances: row check and linear check
     - b norm bounded $2k$ witnesses

Norm Check

The norm check protocol enables the prover to convince the verifier that a vector $\mathbf{f} \in \mathcal{R}^m$ has norm less than b by demonstrating that every component u of $\mathbf{f}$ lies within the interval $[-b, b]$.

Polynomial Construction

This verification is achieved by proving that $g(u) = 0$ for each component, where $g(X)$ is the specifically constructed polynomial:

$$g(X) := X \prod_{i \in [b]} (X - i)(X + i)$$

The zeros of this polynomial g are exactly the set $[-b, b]$, so $g(u) = 0$ if and only if $u \in [-b, b]$.

Proving the Norm Bound

To prove the norm bound of an m-dimensional vector f, we employ the following approach:

- **Multilinear Extension**: Extend f to its multilinear polynomial $\tilde{f}(Y)$ over the Boolean hypercube $\{0,1\}^{\log m}$
- **Polynomial Evaluation**: Since $\tilde{f}(Y)$ agrees with f on the Boolean hypercube $\{0,1\}^{\log m}$, we have $g(\tilde{f}(Y)) = 0$ for all $Y \in \{0,1\}^{\log m}$
- **Sumcheck**: Apply the sumcheck protocol to efficiently verify that

$$\sum_{Y \in \{0,1\}^{\log m}} g(\tilde{f}(Y)) = 0$$

The protocol

Step 1. Evaluation

Prover evaluates $\tilde{f}$ at $0^{\log m}$. ($\tilde{f}$ is denoted by $\mathsf{mle}(\hat{f})$.)

Input: $(\mathbb{x}; \mathbb{w}) := (\mathsf{cm} \in \mathcal{R}_q^{\kappa};\ \vec{\mathbf{f}} \in \mathcal{R}_q^m)$.
Output: $(\mathbb{x}_o; \mathbb{w}_o) := \left((0^{\log m}, \hat{\mathbf{v}} \in \mathcal{R}_q, \mathsf{cm});\ \vec{\mathbf{f}}\right)$.
The protocol $\langle \mathsf{P}(\mathsf{pp}, \mathbb{x}; \mathbb{w}), \mathsf{V}(\mathsf{pp}, \mathbb{x}) \rangle$**:**

1. $\mathsf{P} \to \mathsf{V}$: P sends V the evaluation $\hat{\mathbf{v}} := \mathsf{mle}\left[\hat{\mathbf{f}}\right](0^{\log m})$.
2. V outputs $\mathbb{x}_o := (0^{\log m}, \hat{\mathbf{v}}, \mathsf{cm})$. P outputs $\mathbb{w}_o := \vec{\mathbf{f}}$.

Step 2. Decomposition

Split $\vec{f}$ and its evalution $\hat{\mathbf{v}}$, recompute the commitment y to $\vec{f}_0, \ldots, \vec{f}_{k-1}$.

Input: $\mathbb{x} := (\vec{\mathbf{r}} \in \mathcal{R}_q^{\log m}, \hat{\mathbf{v}} \in \mathcal{R}_q, y \in \mathcal{Y})$ and $\mathbb{w} := \vec{\mathbf{f}} \in \mathcal{R}_q^m$
Output: $[\mathbb{x}_i = (\vec{\mathbf{r}}, \hat{\mathbf{v}}_i, y_i), \mathbb{w}_i = \vec{\mathbf{f}}_i]_{i=0}^{k-1}$
The protocol $\langle \mathsf{P}(\mathsf{pp}, \mathbb{x}; \mathbb{w}), \mathsf{V}(\mathsf{pp}, \mathbb{x}) \rangle$**:**

1. $\mathsf{P} \to \mathsf{V}$: Let $\vec{\mathbf{F}} := (\vec{\mathbf{f}}_0, \ldots, \vec{\mathbf{f}}_{k-1}) := \mathsf{split}_{b,k}(\vec{\mathbf{f}})$. P sends V the values $[y_i, \hat{\mathbf{v}}_i]_{i=0}^{k-1}$ where

$$y_i := \mathcal{L}(\vec{\mathbf{f}}_i)\,, \qquad \hat{\mathbf{v}}_i := \mathsf{mle}\left[\hat{\mathbf{f}}_i\right](\vec{\mathbf{r}})$$

2. V checks that $\sum_{i=0}^{k-1} b^i \cdot y_i \stackrel{?}{=} y$, and $\sum_{i=0}^{k-1} b^i \cdot \hat{\mathbf{v}}_i \stackrel{?}{=} \hat{\mathbf{v}}$.
3. V outputs $[\mathbb{x}_i := (\vec{\mathbf{r}}, \hat{\mathbf{v}}_i, y_i)]_{i=0}^{k-1}$. P outputs $[\mathbb{w}_i := \vec{\mathbf{f}}_i]_{i=0}^{k-1}$.

Step 3. Fold

This step follows the same folding approach as Hypernova, but includes an additional norm check.

Parameters: Strong sampling sets $\mathcal{C}, \mathcal{C}_{\mathsf{small}} \subseteq \mathcal{R}_q$ where $\mathcal{C}$ is defined as in Eq. (31) and $\mathcal{C}_{\mathsf{small}}$ has expansion factor $\|\mathcal{C}_{\mathsf{small}}\|_{\mathsf{op}} \leq c \in \mathbb{N}$ as in (6).
Input: $[\mathbb{x}_i := (\vec{\mathbf{r}}_i, \hat{\mathbf{v}}_i, y_i) \in \mathcal{R}_q^{\log m} \times \mathcal{R}_q \times \mathcal{Y}]_{i=1}^{2k}$ and $[\mathbb{w}_i := \vec{\mathbf{f}}_i \in \mathcal{R}_q^m]_{i=1}^{2k}$
Output: $\mathbb{x}_o := (\vec{\mathbf{r}}_o, \hat{\mathbf{v}}_o, y_o)$, $\mathbb{w}_o := \vec{\mathbf{f}}_o$
The protocol $\langle \mathsf{P}(\mathsf{pp}, \mathbb{x}; \mathbb{w}), \mathsf{V}(\mathsf{pp}, \mathbb{x}) \rangle$ where $\mathbb{x} = [\mathbb{x}_i]_{i=1}^{2k}$ and $\mathbb{w} = [\mathbb{w}_i]_{i=1}^{2k}$:

1. $\mathsf{V} \to \mathsf{P}$: V sends P challenges $[\alpha_i, \mu_i]_{i=1}^{2k} \xleftarrow{\$} (\mathcal{C} \times \mathcal{C})^{2k}$ and $\vec{\beta} \xleftarrow{\$} \mathcal{C}^{\log m}$.
2. $\mathsf{V} \leftrightarrow \mathsf{P}$: P and V run a sum-check protocol for the claim

$$\sum_{\vec{\mathbf{b}} \in \{0,1\}^{\log m}} g(\vec{\mathbf{b}}) = \sum_{i=1}^{2k} \alpha_i \hat{\mathbf{v}}_i \,, \tag{13}$$

where the polynomial $g(\vec{\mathbf{x}}) \in \mathcal{R}_q^{\leq 2b}[X_1, \ldots, X_{\log m}]$ is defined as

$$g(\vec{\mathbf{x}}) := g_{\mathsf{eval}}(\vec{\mathbf{x}}) + g_{\mathsf{norm}}(\vec{\mathbf{x}}) \,, \tag{14}$$

$$g_{\mathsf{eval}}(\vec{\mathbf{x}}) := \sum_{i=1}^{2k} \alpha_i \cdot g_{1,i}(\vec{\mathbf{x}}) \quad \text{where} \quad g_{1,i}(\vec{\mathbf{x}}) := eq(\vec{\mathbf{r}}_i, \vec{\mathbf{x}}) \cdot \mathsf{mle}\left[\hat{\mathbf{f}}_i\right](\vec{\mathbf{x}}) \,, \tag{15}$$

$$g_{\mathsf{norm}}(\vec{\mathbf{x}}) := \sum_{i=1}^{2k} \mu_i \cdot g_{2,i}(\vec{\mathbf{x}}) \quad \text{where} \quad g_{2,i}(\vec{\mathbf{x}}) := eq(\vec{\beta}, \vec{\mathbf{x}}) \cdot \prod_{j=-(b-1)}^{b-1} \left(\mathsf{mle}\left[\hat{\mathbf{f}}_i\right](\vec{\mathbf{x}}) - j\right) \tag{16}$$

The sumcheck protocol reduces checking (13) to checking the evaluation claim $g(\vec{\mathbf{r}}_o) \stackrel{?}{=} s$, where $s \in \mathcal{R}_q$ and $\vec{\mathbf{r}}_o \xleftarrow{\$} \mathcal{C}^{\log m}$ is a sum-check challenge sampled by V.

3. $\mathsf{P} \to \mathsf{V}$: P sends V values $\left[\theta_i := \mathsf{mle}\left[\hat{\mathbf{f}}_i\right](\vec{\mathbf{r}}_o)\right]_{i=1}^{2k}$.
4. V computes $[\mathbf{e}_i := eq(\vec{\mathbf{r}}_i, \vec{\mathbf{r}}_o)]_{i=1}^{2k}$ and $\mathbf{e}^* := eq(\vec{\beta}, \vec{\mathbf{r}}_o)$ and checks that

$$s \stackrel{?}{=} \sum_{i=1}^{2k} \left[\alpha_i \mathbf{e}_i \theta_i + \mu_i \mathbf{e}^* \cdot \prod_{j=1-b}^{b-1} (\theta_i - j)\right] .$$

5. $\mathsf{V} \to \mathsf{P}$: V sends P random challenge $[\rho_i]_{i=1}^{2k} \xleftarrow{\$} \mathcal{C}_{\mathsf{small}}^{2k}$.
6. V outputs $\mathbb{x}_o := (\vec{\mathbf{r}}_o, \hat{\mathbf{v}}_o, y_o)$ where[a]

$$\mathsf{NTT}(\hat{\mathbf{v}}_o) = \sum_{i=1}^{2k} \mathsf{RotSum}(\rho_i, \mathsf{NTT}(\theta_i)) \,, \qquad y_o := \sum_{i=1}^{2k} \rho_i y_i \,.$$

7. P further outputs $\mathbb{w}_o := \vec{\mathbf{f}}_o = \sum_{i=1}^{2k} \rho_i \cdot \vec{\mathbf{f}}_i$.

[a] If $\mathcal{R}_q \cong \mathbb{F}_{q^\tau}^{d/\tau}$ for some $\tau > 1$, the value $\hat{\mathbf{v}}_o$ is replaced with $\hat{V}_o \in \mathcal{R}_q^\tau$ and verifier check is modified as in Eq. (32).

- eq(15) is the Linear check (the same as Hypernova)
- eq(16) is the Norm check
- $\mathsf{NTT}(\hat{\mathbf{v}}_o)$ is to fold the linear evalution $\hat{\mathbf{v}}$

For simplicity, we skipped NTT details from Latticefold.

4. Discussion

Contributions:

1. Introduces the foundational approach for folding in lattice-based cryptography
2. Provides an elegant method for maintaining security constraints during folding operations

Limitations:

1. Only supports folding 2 instances at a time
2. The norm check introduces inefficiency due to the complexity of the polynomial g(X).