Simulation for Quadratic Check:

① Verifier's Randomness - $\{\rho, r, Q, \tau, \gamma, \delta\}$

② Commitments - $\{\Pi[\cdot, k_u]\}_{u \in [t]}$; $\tilde{c}_1, \ldots, \tilde{c}_{\ell}$

$c_1, \ldots, c_{2\ell}$ Depends on proximity

③ Commitment-Randomness - $\{Q_a[\cdot, k_u] : u \in [t]\} \to U_a$

$\{\omega_{k_u} : u \in [t]\} \to c_{k_u}$

$\omega \to PQ$.

④ WitnessRelatedVectors : $\{U_a[\cdot, \cdot, k_u] : u \in [t]\} : a \in \{x, y, z\}$

z

Simulation:

- S picks $\{\rho, r, Q, \tau, \gamma, \delta\}$ uniformly at random
- S picks $U_a[\cdot, \cdot, k_u]$ $\forall u \in [t]$ and $a \in \{x, y, z\}$ uniformly at random such that each plane has columns codewords in L_2. ~~$\forall u$~~ Then S picks z such that $z[j] = 0$ $\forall j \in [m]$. and picks $\omega \xleftarrow{\$} \mathbb{F}$. Computes $cm \leftarrow Com(z, \omega)$.
- S picks $O_x, O_y, O_z \xleftarrow{\$} \mathbb{F}^{pxt}$, then computes
  - (Depends on how the proximity

check is done for (U_x, U_y, U_z) ✓

S picks $\tilde{c}_1, \ldots, \tilde{c}_\ell$ such that ~~it satisfies~~

$$\tilde{c}_{ku} = \sum_{a \in [\ell]} \Lambda_1^T[a, ku]\, \tilde{c}_a \quad \forall\, u \in [t]$$

S picks $\omega_{ku} \xleftarrow{\$} \mathbb{F}$ $\forall\, u \in [t]$, $c_{ku} \leftarrow \mathrm{Com}(\phi[\cdot, ku], \omega_{ku})$

S picks random $c_1, \ldots, c_\ell$ such that

$$c_{ku} = \sum_{a \in [\ell]} \Lambda_4^T[a, ku]\, c_a \quad \forall u \in [t]$$

and $c_m = \sum_{a \in [\ell]} c_a \rho_a$

Simulation for Privacy among the provers:

Maliciously secure DPZK:

If P_3 sends a commitment-outcome cm, then along with cm, it also sends a zero-knowledge Argument of knowledge (ZKAoZ) which says P_3 knows what 'cm' opens to and has binding

property, ~~what~~ with overwhelming property underlying Committed value is same. Since ~~the ar~~ an ~~A~~g Argument of knowledge is ~~is~~ given, so ∃ an extractor $\mathcal{E}$ which can output what 'cm' opens to.

Now we will design a Simulator $\mathcal{S}$ which is going to use ZK simulator S_{ZK} of Graphene, Extractor ~~of~~ $\mathcal{E}$ of the ZKAoK given along with the commitment and S_F, the simulator for the secure MPC used for getting

$$\langle U_x \cdot U_y \rangle^{\xi} \leftarrow \text{Mult}\left(\{\langle U_x \rangle^{\xi}, \langle U_y \rangle^{\xi}\}_{\xi \in [N]}\right)$$

$\mathcal{S}$ calls S_{ZK} and gets the transcript

$$\{\rho, \tau, \Omega, \beta, \delta\}, \{\rho, \tau, \Omega, \tau, \gamma, \delta\}$$

$$\hookrightarrow \{\rho, \tau, \Omega, \beta, \tau, \gamma, \delta\}$$

To

$$\Big\{ \{\Pi[\cdot, ku]\}_{u \in [t]},\ \underbrace{\tilde{c}^{\mathcal{L}}_1, \ldots, \tilde{c}^{\mathcal{L}}_\ell}_{\text{lin}},\ \tilde{c}^{\mathcal{Q}_x}_1, \ldots, \tilde{c}^{\mathcal{Q}_x}_\ell,\ \tilde{c}^{\mathcal{Q}_y}_1, \ldots, \tilde{c}^{\mathcal{Q}_y}_\ell,\ \tilde{c}^{\mathcal{Q}_z}_1, \ldots, \tilde{c}^{\mathcal{Q}_z}_\ell,\ c^{\mathcal{L}}_1, \ldots, c^{\mathcal{L}}_{s+d} \text{ and } c^{\mathcal{Q}}_1, \ldots, c^{\mathcal{Q}}_{2\ell} \Big\}$$

$$\{ O[\cdot, ku], O^x, O^y, O^z \}$$

$$\{ w^{\mathcal{L}}_{ku} : u \in [t] \}, \{ w^{\mathcal{Q}}_{ku} : u \in [t] \}$$

$$\{ w^{\mathcal{L}}, w^{\mathcal{Q}} \}$$

$$\{ U[\cdot, \cdot, ku], U_x[\cdot, \cdot, ku], U_y[\cdot, \cdot, ku], U_z[\cdot, \cdot, ku] \quad \forall u \in [t] \}$$

$$\{ z^{\mathcal{L}}, z^{\mathcal{Q}} \}$$

Provers in the corrupt set T send $\langle \mathcal{U}^{com} \rangle^{\Sigma}, \langle \mathcal{U}_x^{com} \rangle^{\Sigma}, \langle \mathcal{U}_y^{com} \rangle^{\Sigma}, \langle \mathcal{U}_z^{com} \rangle^{\Sigma}$ in a broadcast channel and so Simulator $\mathcal{S}$ receives these commitments with ZKAoK. $\mathcal{S}$ calls the extractor and extracts U, U_x, U_y, U_z. Corresponding to the ρ, $\mathcal{A}$ sends

$\langle \tilde{C}^L \rangle^\xi, \ldots, \langle \tilde{C}_l^L \rangle^\xi$, $\langle \tilde{C}_a^{\otimes} \rangle^\xi, \ldots, \langle \tilde{C}_{la}^{\otimes} \rangle^\xi$, $a \in \{x, y, z\}$

with the ZKAoK. $\mathcal{S}$ uses the extractor $\mathcal{E}$ and outputs $\langle U[\cdot,1] \rangle^\xi, \ldots, \langle U[\cdot,l] \rangle^\xi, \langle U_a[\cdot,1] \rangle^\xi, \ldots \langle U_a[\cdot,l] \rangle^\xi \quad \forall a \in \{x,y,z\}$ and computes the complete $\langle U \rangle^\xi, \langle U_a \rangle^\xi, \forall a \in \{x,y,z\}$

$\mathcal{S}$ computes the commitments for honest parties using the transcript generated by S_{ZK}.

$\mathcal{S}$ calls the simulator S_p of the ~~si~~ of the secure multiplication protocol, S_p extracts the input of the MPC for the corrupt parties. If the extracted input by S_p is same as $\langle U_x \rangle^\xi$ and $\langle U_y \rangle^\xi$ then $\mathcal{S}$ gives the output ~~of~~ on behalf of the ideal functionality which is chosen at random $\langle U_x \cdot U_y \rangle^\xi \quad \forall \xi \in T$.

$\mathcal{S}$ computes P^2 and $P^{\otimes}$, matrices for the Linear check and Quadratic check

S receives messages $\langle c_0^{\Sigma}, \langle c_1 \rangle^{\Sigma} \cdots \langle c_{s+t} \rangle^{\Sigma}, \langle d_0 \rangle^{\Sigma}$ (L) and $\langle c_0^{\Theta} \rangle^{\Sigma}, \ldots \langle c_{2t}^{\Theta} \rangle^{\Sigma}$ along with ZKAoK and it uses the extractor $\mathcal{E}$ and gets $\langle \bar{P}^{L} \rangle^{\Sigma}$ and $\langle \bar{P}^{\Theta} \rangle^{\Sigma}$ and check if $\langle \bar{P}^{L} \rangle^{\Sigma}[j,k] = \sum_{i \in [p]} R^{i}(\alpha_j, \gamma_k) \langle V \rangle^{\Sigma}[j,k]$ and $\langle \bar{P}^{\Theta} \rangle^{\Sigma}[j,k] = \sum_i r_i [\langle U_x \cdot U_y \rangle^{\Sigma}[i,j,k] - \langle U_z \rangle^{\Sigma}[i,j,k]]$

If Yes, then S cooks up the messages for honest parties using the transcript generated by S_{ZK}, if $\mathcal{F}_{DPZK}(\langle w_i \rangle^{\Sigma}) \to 1$ such as

$\langle c_k \rangle^{\Sigma} = \mathrm{Sh}(c_k = \sum_{\Sigma \in T} \langle c_k \rangle^{\Sigma})$ for both

Linear and quadratic and generates a transcript which is accepting.

else, if No, then S picks random values on behalf of honest parties. which is indistinguishable from a correct execution

(Note that to pick these values uniformly at random, S picks some random value and commits

to that- and generates a correct

ZKAoK & corresponding to those values).