Linearly Homomorphic Secret Sharing and Multi-Party Computation with Unanimously Verifiable Deletion

Yilei Chen* Liheng Ji† Han Luo‡

September 19, 2026

Abstract

Certified deletion enables a party to prove that it has erased the sensitive information contained in a quantum state. Bartusek and Raizes (CRYPTO 2024) gave the first secret-sharing scheme with privately verifiable deletion. Subsequently, Katz and Sela (EUROCRYPT 2025) constructed secret-sharing schemes with publicly verifiable deletion under computational assumptions. Constructing such a publicly verifiable scheme without cryptographic assumptions remains open.

In this work, we introduce an intermediate notion called *unanimously verifiable deletion*. While private verification relies on the dealer and public verification allows any third party to verify deletion, unanimous verification assigns the verification responsibility to the parties in the secret-sharing scheme themselves. We construct an information-theoretically secure *linearly homomorphic threshold secret-sharing batch scheme with adaptive unanimously verifiable deletion*. In particular, to obtain its linear-homomorphic property, we exploit a *linearly homomorphic structure of BB84-type states* that enables linear transformations on encoded values while preserving certified deletion. More generally, this structure can be used to upgrade certain primitives with certified deletion to their linearly homomorphic counterparts.

As another major contribution, we demonstrate an application of this batch scheme by constructing *outsourced multi-party computation* (MPC) protocols for arbitrary arithmetic circuits. Our outsourced MPC has *semi-honest classical clients* and *malicious quantum servers*, achieves *adaptive unanimously verifiable deletion security* in the *trusted-preprocessing* and *security-with-abort* setting against a *rushing* adversary, and supports *public output reconstruction*. The adversary may adaptively corrupt up to $n-1$ clients. For the servers, during the online execution, at most t corrupted servers may remain undeleted and at least $t+1$ honest servers must remain undeleted, for any $t < n/2$. After successful finalization, all servers may be corrupted without revealing additional information about the remaining clients' inputs beyond the public output and the corrupted clients' inputs.

1 Introduction

Certified deletion, a concept first formalized by Broadbent and Islam [BI20], allows a party to provably delete sensitive information in the quantum state it holds. By default, certified dele-

*IIIS, Tsinghua University and Shanghai Qi Zhi Institute. chenyilei@mail.tsinghua.edu.cn. Supported by Shanghai Qi Zhi Institute Innovation Program.

†IIIS, Tsinghua University and Shanghai Qi Zhi Institute. jlh23@mails.tsinghua.edu.cn.

‡IIIS, Tsinghua University. luohcs@gmail.com.

tion refers to privately verifiable deletion, meaning that only a certain party (usually the one who prepares the state) can verify that deletion has occurred. Publicly verifiable deletion extends this concept by enabling any (third) party to verify the deletion certificate, enhancing transparency and applicability in distributed settings. So far, many cryptographic primitives with privately (e.g. [Unr15, BGK$^+$24, BK23, BR24, HKM$^+$24, HMNY21, HMNY22, KN22, AKN$^+$23, APV23, AHH24, MPY24, PWYZ24, KMY25, CGJL25, KNP25, KNY21, KT23, AK25, KNP26]) or publicly (e.g. [BKM$^+$23, KNY23, Por23, BKP23]) verifiable deletion have been defined and constructed.

A (t, n)-threshold secret sharing scheme [Sha79] allows a dealer to split a secret s into n shares, such that any $t+1$ or more shares can be combined to reconstruct s, but any t or less shares reveal no information about s. In the classical world, we usually assume that the adversary can only corrupt at most t shares. However, quantum allows us to think about a more flexible assumption of corruption using certifiable deletion: even if the adversary has corrupted t shares, it can still continue corruption after it provably deletes some corrupted shares, with the guarantee that the number of undeleted corrupted shares never exceeds t. Bartusek and Raizes [BR24] initiate the study of secret sharing with certified deletion. They formally define two security notions, *no-signaling security* and *adaptive security*, and provide a construction for each of them. Later, Katz and Sela [KS25] manage to construct secret sharing schemes with publicly verifiable deletion, assuming the quantum hardness of LWE or existence of quantum-secure OWF.

Currently, constructions of secret sharing with publicly verifiable deletion without any assumption are not known. However, we can take a step back: instead of allowing any third party to verify the deletion certificate, we let the parties in the scheme to do verification together. Concretely, after some party deletes its share and produces a deletion certificate, every other party will verify the validity of this certificate, and accepts if and only if no one thinks this certificate is invalid. We name this concept by *unanimously verifiable deletion*. This notion is stronger than privately verifiable deletion, and may be easier to achieve than publicly verifiable deletion.

Another major focus of this work is secure multi-party computation (MPC), or precisely, outsourced MPC. MPC with certified deletion can be viewed as the natural extension of secret sharing with certified deletion from storage to computation. If secret sharing with certified deletion provides a storage service in which a provider can verifiably erase the stored secret, then MPC with certified deletion provides an analogous computation service in which the servers can verifiably erase the private information used during the computation. This allows the cumulative number of corrupted servers to exceed the usual corruption threshold during the execution, provided that sufficiently many previously exposed servers have deleted their local states and the number of corrupted but undeleted servers remains below the threshold. Moreover, after the computation has been completed, all the servers may be corrupted. Such security is not provided by classical outsourced MPC.

Despite this close conceptual connection, there remains a substantial gap between secret sharing with certified deletion and its use in MPC. Existing secret-sharing schemes with certified deletion are primarily designed for storing and reconstructing secrets, and do not directly support computation over the shared values. In particular, shares of different secrets cannot in general be combined through the homomorphic operations required for MPC while retaining their certified-deletion guarantees. Bridging this gap therefore requires a secret-sharing scheme that supports both computation and certified deletion.

Previous discussions led to the following questions:

1. Is there a secret-sharing scheme that supports *unanimously verifiable deletion* without relying on any cryptographic assumptions?

2. Can secret sharing with certified deletion be equipped with a homomorphic structure that supports computation over shared secrets? If so, can they be used to realize MPC with certified deletion?

1.1 Main Results

In this paper, we answer both questions affirmatively. Our main contributions are as follows.

Linearly homomorphic secret-sharing batch with unanimously verifiable deletion. Our first contribution is to introduce the notion of the threshold linearly homomorphic secret-sharing batch with adaptive unanimously verifiable deletion security and to give an information-theoretically secure construction.

Theorem 1 (Informal)**.** *For every $n \geq 2$ and every $t \in [1, n-1]$, there exists a (t,n)-threshold*[1] *linearly homomorphic secret-sharing batch scheme with adaptive unanimously verifiable deletion security.*

Here, by a "linearly homomorphic batch", we mean that multiple secrets are jointly shared under common setup information so that linear operations can be performed among these sharings. This batch structure will play an essential role in our MPC construction.

The main technical ingredient underlying our construction is a simple linear-homomorphic property of BB84-type states with a common basis string. In particular, we observe that when two BB84-type states use the same basis string, a simple local unitary preserves their BB84-type form while implementing a linear update on their labels. Crucially for our purposes, this allows both the encoded values and the corresponding deletion-verification key to be updated consistently. We believe that this observation is of independent interest, as it provides a general approach for upgrading certain cryptographic primitives with certified deletion to their linearly homomorphic counterparts. As an illustration, Appendix A gives a linearly homomorphic symmetric-key encryption scheme with certified deletion.

Out-sourced MPC with unanimously verifiable deletion. Our second main contribution is to introduce adaptive unanimously verifiable deletion security for outsourced MPC with semi-honest classical client and malicious quantum server, and to construct such a protocol for arbitrary arithmetic circuits.

Theorem 2 (Informal)**.** *For every $t < n/2$ and every polynomial-size arithmetic circuit $C : \mathbb{Z}_q^n \to \mathbb{Z}_q^*$ over a prime field $\mathbb{Z}_q$, there exists an outsourced MPC protocol with* semi-honest classical clients *and* malicious quantum servers *that efficiently computes C and achieves* adaptive unanimously verifiable deletion security *with public output reconstruction in the* trusted-preprocessing *and* security-with-abort *setting against a* rushing *adversary. The adversary may adaptively corrupt up to $n-1$ clients. For the servers, throughout the online execution,*

[1] Throughout this paper, a (t,n)-threshold secret-sharing scheme means a $(t+1)$-out-of-n scheme; this convention differs slightly from that in [BR24, KS25].

Scheme	Security	Underlying Assumption[a]	Verifiable Deletion	MPC
[BR24, Sec. 6]	no-signaling	information-theoretic	privately verifiable	×
[BR24, Sec. 7]	adaptive	information-theoretic	privately verifiable	×
[KS25, Secs. 4, 5]	adaptive	LWE or OWF[b]	publicly verifiable	×
[KS25, App. D]	no-signaling	subexponentially secure OWF	publicly verifiable	×
ours	adaptive	information-theoretic	unanimously verifiable	✓

[a] All computational assumptions in this table are understood to be secure against quantum adversaries.
[b] The construction based on LWE is more efficient than the one based on OWF.
Remark. No-signaling security and adaptive security are incomparable [BR24, KS25, BN24]. Because both unanimous verification of deletion certificates and MPC evaluation inherently require communication among multiple parties, it is unclear how to formulate a meaningful no-signaling security in our setting. We therefore restrict attention to adaptive security.

Table 1: Comparison of existing secret-sharing schemes with certified deletion.

(1) the number of undeleted corrupted servers is at most t; and

(2) the number of undeleted honest servers is at least $t+1$.

After successful finalization, all servers may be corrupted without revealing any additional information about the uncorrupted clients' inputs beyond the public output and the corrupted clients' inputs.

The cumulative number of corrupted servers may therefore exceed the usual corruption threshold during the execution, as long as sufficiently many servers are verifiably deleted before further corruptions and the two conditions above continue to hold. When no server deletes its state, Condition (1) reduces to the standard bound of at most t corrupted servers, while Condition (2) follows from $n \geq 2t+1$. More importantly, after all the computation and deletion procedures are done, the security continues to hold even under the subsequent corruption of all servers.

For a detailed summary of the MPC setting, refer to Remark 22 in Section 3.2. For comparison with prior work on secret sharing with certified deletion, Table 1 summarizes the existing n-party secret sharing schemes with certified deletion.

1.2 Technical Overview of Constructions

A linearly homomorphic structure of BB84-type states. We begin with the main technical observation underlying our constructions. Let q be a prime and let QFT denote the quantum Fourier transform over $\mathbb{Z}_q$. For $\mathbf{x} \in \mathbb{Z}_q^m$ and $\boldsymbol{\theta} \in \{0,1\}^m$, define the BB84-type state $|\mathbf{x}^{\boldsymbol{\theta}}\rangle := \bigotimes_{j=1}^m \mathsf{QFT}^{\theta[j]}|x[j]\rangle$, where $\mathsf{QFT}^0 = \mathbf{I}$ and $\mathsf{QFT}^1 = \mathsf{QFT}$.

For any $c \in \mathbb{Z}_q$, consider the local unitary $\mathbf{U}_c^m$ defined by

$$\forall \mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^m,\ \mathbf{U}_c^m|\mathbf{a}\rangle|\mathbf{b}\rangle := |\mathbf{a}\rangle|c\mathbf{a}+\mathbf{b}\rangle. \tag{1}$$

Suppose that two BB84-type states $|\mathbf{x}_1^{\boldsymbol{\theta}}\rangle$ and $|\mathbf{x}_2^{\boldsymbol{\theta}}\rangle$ share the same basis string $\boldsymbol{\theta}$. Then applying $\mathbf{U}_c^m$ preserves their BB84-type form: the resulting state is $|\widetilde{\mathbf{x}}_1^{\boldsymbol{\theta}}\rangle|\widetilde{\mathbf{x}}_2^{\boldsymbol{\theta}}\rangle$, where, for every $j \in [m]$,

$$(\widetilde{x}_1[j], \widetilde{x}_2[j]) := \begin{cases} (x_1[j], cx_1[j]+x_2[j]), & \theta[j]=0, \\ (x_1[j]-cx_2[j], x_2[j]), & \theta[j]=1. \end{cases}$$

We formalize this property in Theorem 23.

This linearly homomorphic structure is particularly useful for certified deletion based on BB84-type states. Usually, for a BB84 state $|\mathbf{x}^{\boldsymbol{\theta}}\rangle$, the positions with $\theta[j] = 0$ carry the effective information (referred to as evaluation points), whereas the positions with $\theta[j] = 1$ serve as hidden checking positions (referred to as checking points). To perform certified deletion on $|\mathbf{x}^{\boldsymbol{\theta}}\rangle$, the holder of the state should measure the entire state in the Fourier basis and send the outcome $\mathbf{y}$ to a verifier. The verifier, who knows the hidden basis string $\boldsymbol{\theta}$ and the checking labels $\{x[j]\}_{j:\theta[j]=1}$, accepts only if $y[j] = x[j]$ for every j with $\theta[j] = 1$. Since the holder does not know which positions are checked, producing a valid certificate forces it to measure all positions and thereby erase the effective information stored at the positions with $\theta[j] = 0$.

Our linearly homomorphic structure makes this paradigm compatible with linear evaluation. For illustration, suppose that two BB84-type states $|\mathbf{x}_1^{\boldsymbol{\theta}}\rangle$ and $|\mathbf{x}_2^{\boldsymbol{\theta}}\rangle$ encode, respectively, the values $s_1 := \sum_{j:\theta[j]=0} x_1[j]$ and $s_2 := \sum_{j:\theta[j]=0} x_2[j]$, with verification keys $\mathsf{vk}_1 := \{(j, x_1[j])\}_{j:\theta[j]=1}$ and $\mathsf{vk}_2 := \{(j, x_2[j])\}_{j:\theta[j]=1}$. After applying $\mathbf{U}_c^m$ to $|\mathbf{x}_1^{\boldsymbol{\theta}}\rangle|\mathbf{x}_2^{\boldsymbol{\theta}}\rangle$, the resulting states encode

$$\widetilde{s}_1 := \sum_{j:\theta[j]=0} \widetilde{x}_1[j] = s_1 \qquad \text{and} \qquad \widetilde{s}_2 := \sum_{j:\theta[j]=0} \widetilde{x}_2[j] = cs_1 + s_2.$$

Meanwhile, at every checking position j with $\theta[j] = 1$, the labels become $\widetilde{x}_1[j] = x_1[j] - cx_2[j]$ and $\widetilde{x}_2[j] = x_2[j]$. Hence, the verifier can update the verification keys classically as

$$\widetilde{\mathsf{vk}}_1 := \{(j, x_1[j] - cx_2[j])\}_{j:\theta[j]=1} \qquad \text{and} \qquad \widetilde{\mathsf{vk}}_2 := \mathsf{vk}_2.$$

Thus, the transformed states encode s_1 and $cs_1 + s_2$, respectively, while remaining directly deletable and verifiable under the updated verification keys. This provides a general way to upgrade BB84-based primitives with certified deletion to linearly homomorphic variants. For example, Appendix A gives a linearly homomorphic symmetric-key encryption scheme with certified deletion. Next, we show how to apply this technique to Bartusek and Raizes' secret sharing scheme with adaptive certified deletion [BR24, Section 7].

Linear homomorphism of [BR24]'s secret sharing with certified deletion. We now apply the linearly homomorphic structure above to the threshold secret sharing scheme with adaptive certified deletion of Bartusek and Raizes [BR24, Section 7]. We begin by recalling the definition of adaptive security for (t, n)-secret sharing with certified deletion.

Definition 3 (Informal, Adaptive Security of (t, n)-Secret Sharing with Certified Deletion)**.** *A (t, n)-secret-sharing scheme has adaptive certified deletion security if, for any adversary $\mathcal{A}$ and any secret $s \in \mathbb{Z}_q$, $\mathcal{A}$ cannot learn s through the following experiment:*

1. *The challenger splits s into n shares $\mathsf{sh}_1, \mathsf{sh}_2, \ldots, \mathsf{sh}_n$. Then, it initializes two empty sets $\mathcal{CS}, \mathcal{DS}$.*
2. *For as many rounds as the adversary $\mathcal{A}$ like, it chooses one of the two following operations:*
   - *Corrupt a party: $\mathcal{A}$ chooses an $i \notin \mathcal{CS}$. The challenger $\mathcal{C}$ first updates $\mathcal{CS} \leftarrow \mathcal{CS} \cup \{i\}$. If $|\mathcal{CS} \backslash \mathcal{DS}| > t$, it aborts immediately. Otherwise, $\mathcal{C}$ sends sh_i to $\mathcal{A}$ if $i \notin \mathcal{DS}$.*

- *Delete a party's share: $\mathcal{A}$ chooses an $i \notin \mathcal{DS}$. If $i \in \mathcal{CS}$, then $\mathcal{A}$ submits a certificate cert_i for deleting sh_i. If $\mathcal{C}$ thinks cert_i is valid, it updates $\mathcal{DS} \leftarrow \mathcal{DS} \cup \{i\}$; otherwise, it aborts immediately. If $i \notin \mathcal{CS}$, then $\mathcal{C}$ honestly deletes sh_i and updates $\mathcal{DS} \leftarrow \mathcal{DS} \cup \{i\}$.*

3. $\mathcal{A}$ outputs its view when it is done.

We remark that compared to the original definition in [BR24], ours includes the scenario where an honest party may also delete its share. Currently, this does not affect the adversary's view. But later, observant readers will find some difference in the case of unanimously verifiable deletion.

Now, we introduce the adaptively secure scheme of [BR24]. To split a secret $s \in \mathbb{Z}_q$, a trusted party $\mathcal{T}$ (dealer) samples a $\mathbb{Z}_q$-polynomial f with $f(0) = s$. Then, for each party $i \in [n]$, $\mathcal{T}$ sends $\mathcal{P}_i$ a quantum state $\mathsf{sh}_i := |\boldsymbol{\psi}_i\rangle := \bigotimes_{j=1}^{m} |\psi_{i,j}\rangle$, where each $|\psi_{i,j}\rangle$ is either an evaluation of f on the computational basis (an evaluation point), or a random number on the QFT basis (a checking point). Finally, $\mathcal{T}$ stores the positions and values of the checking points as the verification key. In the following we present a concrete example with $n = 3$.

$$
\begin{array}{lccccccccc}
|\boldsymbol{\psi}_1\rangle = & |f(1)\rangle & \otimes & |f(2)\rangle & \otimes & \mathsf{QFT}|r_{1,3}\rangle & \otimes & \ldots & \otimes & |f(m)\rangle \\
\mathsf{vk}_1 = & * & & * & & (3, r_{1,3}) & & \ldots & & * \\
|\boldsymbol{\psi}_2\rangle = & |f(m+1)\rangle & \otimes & \mathsf{QFT}|r_{2,2}\rangle & \otimes & \mathsf{QFT}|r_{2,3}\rangle & \otimes & \ldots & \otimes & |f(2m)\rangle \\
\mathsf{vk}_2 = & * & & (2, r_{2,2}) & & (3, r_{2,3}) & & \ldots & & * \\
|\boldsymbol{\psi}_3\rangle = & \mathsf{QFT}|r_{3,1}\rangle & \otimes & |f(2m+2)\rangle & \otimes & |f(2m+3)\rangle & \otimes & \ldots & \otimes & \mathsf{QFT}|r_{3,m}\rangle \\
\mathsf{vk}_3 = & (1, r_{3,1}) & & * & & * & & \ldots & & (m, r_{3,m})
\end{array}
\tag{2}
$$

For certified deletion, a party $\mathcal{P}_i$ measures its share on the QFT basis and gives the measurement result to $\mathcal{T}$, who uses vk_i to check whether the values on checking points are correct. For reconstruction, the parties measure their shares on the computational basis, gather the results and perform polynomial interpolation with error correction to recover f. By tuning the number of evaluation and checking points in each share, as well as the degree of the polynomial carefully, we can guarantee that (1) any $\geq t+1$ shares is enough to reconstruct f, while (2) any $\leq t$ shares are not enough to reconstruct f. The intuition of adaptive security is that, the adversary $\mathcal{A}$ can never get more than t shares' measuring results on the computational basis. By the property of polynomials, $s = f(0)$ keeps hiding to $\mathcal{A}$.

Since polynomials are additive, we can establish linear homomorphism in two such shares with the same checking points. As an example, assume there is another sharing for with secret s' as follows (we only write the first share for simplicity):

$$
\begin{array}{lccccccccc}
|\boldsymbol{\psi}'_1\rangle = & |f'(1)\rangle & \otimes & |f'(2)\rangle & \otimes & \mathsf{QFT}|r'_{1,3}\rangle & \otimes & \ldots & \otimes & |f'(m)\rangle \\
\mathsf{vk}'_1 = & * & & * & & (3, r'_{1,3}) & & \ldots & & *
\end{array}
\tag{3}
$$

where $f'(0) = s'$. By applying $\mathbf{U}_c^m$ (see Eqn (1)) to $|\boldsymbol{\psi}_1\rangle|\boldsymbol{\psi}'_1\rangle$, and updating vk and vk' accordingly,

we convert them into

$$\begin{array}{lccccccccc} |\tilde{\psi}_1\rangle = & |f(1)\rangle & \otimes & |f(2)\rangle & \otimes & \mathsf{QFT}|r_{1,3} - cr'_{1,3}\rangle & \otimes & \dots & \otimes & |f(m)\rangle \\ \mathsf{vk}_1 = & * & & * & & (3, r_{1,3} - cr'_{1,3}) & & \dots & & * \end{array}$$

$$\begin{array}{lccccccccc} |\tilde{\psi}'_1\rangle = & |f'(1) + cf(1)\rangle & \otimes & |f'(2) + cf(2)\rangle & \otimes & \mathsf{QFT}|r'_{1,3}\rangle & \otimes & \dots & \otimes & |f'(m) + cf(m)\rangle \\ \mathsf{vk}'_1 = & * & & * & & (3, r'_{1,3}) & & \dots & & * \end{array} \tag{4}$$

We have the former is still a sharing with secret s, and the latter becomes a sharing with secret $f'(0) + cf(0) = s' + cs$.

Add a public bias. To simplify scalar additions in the MPC protocol, we make a slight modification to the secret sharing scheme above. To share a secret s, instead of sampling a polynomial with $f(0) = s$, the trusted party first samples a random bias $\sigma \leftarrow \mathbb{Z}_q$, and then samples a polynomial f with $f(0) = s - \sigma$. Then for each $i \in [n]$, it samples $|\psi_i\rangle$ as in Eqn (2), and set $\mathsf{sh}_i := (|\psi_i\rangle, \sigma)$. Note that each secret has its own independently sampled bias. In particular, for another secret $s' \in \mathbb{Z}_q$, the trusted party should sample a fresh bias $\sigma' \leftarrow \mathbb{Z}_q$, samples the corresponding polynomial f' with $f'(0) = s' - \sigma'$, and sets $\mathsf{sh}'_i := (|\psi'_i\rangle, \sigma')$ for every $i \in [n]$. This modification preserves the linear homomorphism among sharings. Moreover, it makes adding a scalar to a shared secret particularly simple: to add a scalar c to the secret value s, it suffices to replace σ by $\sigma + c$.

So far, building on the secret-sharing scheme of [BR24], we have established how to perform linear operations over such sharings. Nevertheless, this alone is still insufficient for obtaining an MPC protocol. On the one hand, multiplication is not captured by such linear operations and remains a central obstacle. On the other hand, if one directly applies such a secret-sharing scheme to outsourced MPC, different input clients $\mathcal{C}_1, \dots, \mathcal{C}_n$ would act as dealers for their respective inputs; however, sharings generated independently by different dealers do not automatically admit the linearity property, since their checking positions, or equivalently the underlying basis strings, may be different. To overcome these obstacles, we need more powerful tools and techniques. We begin by introducing unanimously verifiable deletion in the following.

Linearly homomorphic secret-sharing batch with unanimously verifiable deletion. We now introduce the concept of unanimously verifiable deletion. Instead of asking help from a trusted third party, we want the parties to check the validity of a deletion certificate by themselves. The idea is to divide the checking position of each share into n parts. Concretely, for each $i_1, i_2 \in [n]$, $\mathcal{P}_{i_1}$ knows ℓ checking points on $\mathcal{P}_{i_2}$'s share, which is denoted by vk_{i_1,i_2}. We guarantee that each $\mathcal{P}_{i_1}$ knows totally different checking points on $\mathcal{P}_{i_2}$'s share, that is, $\{\mathsf{vk}_{i_1,i_2}\}_{i_1 \in [n]}$ are mutually disjoint sets for a certain i_2. In the following we give a concrete example with $n = 3, m = 7, \ell = 1$. We color

$\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$'s possessions in red, green and blue respectively.

$$
\begin{array}{lccccccccccccc}
|\psi_1\rangle= & |f(1)\rangle & \otimes & |f(2)\rangle & \otimes & \mathsf{QFT}|r_{1,3}\rangle & \otimes & \mathsf{QFT}|r_{1,4}\rangle & \otimes & |f(5)\rangle & \otimes & \mathsf{QFT}|r_{1,6}\rangle & \otimes & |f(7)\rangle \\
 & * & & * & & \mathsf{vk}_{1,1}{=}(3,r_{1,3}) & & \mathsf{vk}_{2,1}{=}(4,r_{1,4}) & & * & & \mathsf{vk}_{3,1}{=}(6,r_{1,6}) & & * \\
|\psi_2\rangle= & \mathsf{QFT}|r_{2,1}\rangle & \otimes & |f(9)\rangle & \otimes & |f(10)\rangle & \otimes & \mathsf{QFT}|r_{2,4}\rangle & \otimes & |f(12)\rangle & \otimes & \mathsf{QFT}|r_{2,6}\rangle & \otimes & |f(14)\rangle \\
 & \mathsf{vk}_{1,2}{=}(1,r_{2,1}) & & * & & * & & \mathsf{vk}_{3,2}{=}(4,r_{2,4}) & & * & & \mathsf{vk}_{2,2}{=}(6,r_{2,6}) & & * \\
|\psi_3\rangle= & |f(15)\rangle & \otimes & \mathsf{QFT}|r_{3,2}\rangle & \otimes & \mathsf{QFT}|r_{3,3}\rangle & \otimes & |f(18)\rangle & \otimes & \mathsf{QFT}|r_{3,5}\rangle & \otimes & |f(20)\rangle & \otimes & |f(21)\rangle \\
 & * & & \mathsf{vk}_{2,3}{=}(2,r_{3,2}) & & \mathsf{vk}_{3,3}{=}(3,r_{3,3}) & & * & & \mathsf{vk}_{1,3}{=}(5,r_{3,5}) & & * & & *
\end{array}
$$

Consider, for example, that $\mathcal{P}_1$ is told to delete its share. It measures $|\psi_1\rangle$ on the QFT basis, obtains a deletion certificate $\mathsf{cert}_1 \in \mathbb{Z}_q^m$, and broadcasts it. Then, $\mathcal{P}_2$ uses $\mathsf{vk}_{2,1}$ to check whether $\mathsf{cert}_1[4] = r_{1,4}$, while $\mathcal{P}_3$ uses $\mathsf{vk}_{3,1}$ to check whether $\mathsf{cert}_1[6] = r_{1,6}$. Since $\mathcal{P}_1$ does not know either $\mathsf{vk}_{2,1}$ or $\mathsf{vk}_{3,1}$, passing these hidden checks forces it to measure $|\psi_1\rangle$ on the QFT basis.

We then introduce the concept of the linearly homomorphic secret-sharing batch with unanimously verifiable deletion. This notion captures the scenario of a set of secret sharings with unanimously verifiable deletion having the same evaluation and checking positions, and thus are linearly homomorphic. Concretely, assume now we want to construct a size-M sharing batch with secrets $s_1, \ldots, s_M$. The trusted third party $\mathcal{T}$ first runs a Setup algorithm that samples n^2 master verification keys $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}$, where $\mathsf{mvk}_{i_1,i_2} \subseteq [m]$ denotes the indices of checking points on $\mathcal{P}_{i_2}$'s share known to $\mathcal{P}_{i_1}$. Then, for $\iota \in [M]$, $\mathcal{T}$ splits each secret s_ι into n shares according to $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}$. To make things clear, we list the possessions of $\mathcal{P}_{i_1}$ as follows:

- For each $\iota \in [M]$, a share $\mathsf{sh}_{i_1,\iota} := (|\psi_{i_1,\iota}\rangle, \sigma_\iota)$ for s_ι.
- For each $i_2 \in [n]$, a master verification key mvk_{i_1,i_2} for $\mathcal{P}_{i_2}$.
- For each $i_2 \in [n]$ and $\iota \in [M]$, a verification key $\mathsf{vk}_{i_1,i_2,\iota}$ for $\mathcal{P}_{i_2}$'s share of s_ι.

We end this part by giving a concrete example of such secret-sharing batch scheme with $n = 3, m = 7, \ell = 1$. For all $\iota \in [M]$, we have

$$
\begin{array}{lccccccccccccc}
|\psi_{1,\iota}\rangle= & |f_\iota(1)\rangle & \otimes & |f_\iota(2)\rangle & \otimes & \mathsf{QFT}|r_{1,3,\iota}\rangle & \otimes & \mathsf{QFT}|r_{1,4,\iota}\rangle & \otimes & |f_\iota(5)\rangle & \otimes & \mathsf{QFT}|r_{1,6,\iota}\rangle & \otimes & |f_\iota(7)\rangle \\
 & * & & * & & \mathsf{vk}_{1,1,\iota}{=}r_{1,3,\iota} & & \mathsf{vk}_{2,1,\iota}{=}r_{1,4,\iota} & & * & & \mathsf{vk}_{3,1,\iota}{=}r_{1,6,\iota} & & * \\
 & * & & * & & \mathsf{mvk}_{1,1}{=}3 & & \mathsf{mvk}_{2,1}{=}4 & & * & & \mathsf{mvk}_{3,1}{=}6 & & * \\
|\psi_{2,\iota}\rangle= & \mathsf{QFT}|r_{2,1,\iota}\rangle & \otimes & |f_\iota(9)\rangle & \otimes & |f_\iota(10)\rangle & \otimes & \mathsf{QFT}|r_{2,4,\iota}\rangle & \otimes & |f_\iota(12)\rangle & \otimes & \mathsf{QFT}|r_{2,6,\iota}\rangle & \otimes & |f_\iota(14)\rangle \\
 & \mathsf{vk}_{1,2,\iota}{=}r_{2,1,\iota} & & * & & * & & \mathsf{vk}_{3,2,\iota}{=}r_{2,4,\iota} & & * & & \mathsf{vk}_{2,2,\iota}{=}r_{2,6,\iota} & & * \\
 & \mathsf{mvk}_{1,2}{=}1 & & * & & * & & \mathsf{mvk}_{3,2}{=}4 & & * & & \mathsf{mvk}_{2,2}{=}6 & & * \\
|\psi_{3,\iota}\rangle= & |f_\iota(15)\rangle & \otimes & \mathsf{QFT}|r_{3,2,\iota}\rangle & \otimes & \mathsf{QFT}|r_{3,3,\iota}\rangle & \otimes & |f_\iota(18)\rangle & \otimes & \mathsf{QFT}|r_{3,5,\iota}\rangle & \otimes & |f_\iota(20)\rangle & \otimes & |f_\iota(21)\rangle \\
 & * & & \mathsf{vk}_{2,3,\iota}{=}r_{3,2,\iota} & & \mathsf{vk}_{3,3,\iota}{=}r_{3,3,\iota} & & * & & \mathsf{vk}_{1,3,\iota}{=}r_{3,5,\iota} & & * & & * \\
 & * & & \mathsf{mvk}_{2,3}{=}2 & & \mathsf{mvk}_{3,3}{=}3 & & * & & \mathsf{mvk}_{1,3}{=}5 & & * & & *
\end{array}
$$

where f_ι is a polynomial that $f_\iota(0) = s_\iota - \sigma_\iota$. Note that the notation here differs slightly from that used in the construction of Section 4. For instance, in Section 4, we write we write $\mathsf{vk}_{i_1,i_2,\iota} := \{(j, r_{i_2,j,\iota})\}_{j\in\mathsf{mvk}_{i_1,i_2}}$, explicitly recording each checking position together with its value.

MPC with unanimously verifiable deletion for arithmetic circuits. We now apply the techniques above to construct MPC protocols for arithmetic circuits. We work in the outsourced setting with semi-honest classical clients and malicious quantum servers. The clients only provide their inputs through classical masked messages and then retire, while the servers hold and process the quantum secret-sharing states and carry out the circuit evaluation, deletion, and reconstruction procedures.

Example 1: a single addition. We first show how to construct an MPC protocol that computes

$$F_1(x_1, x_2) := x_1 + x_2,$$

which serves as a basic example of how to compute linear functions. (The cases of scalar multiplication and scalar addition can be handled similarly.)

For every $s \in \mathbb{Z}_q$, we use $[\![s]\!]_\iota$ to denote the ι-th sharing in the secret-sharing batch, and use $\langle s \rangle$ to denote a classical n-out-of-n robust-openable sharing of s. We use "$\Leftarrow$" to indicate a homomorphic update of a sharing, where the secret encoded by the sharing is changed according to the expression on the right-hand side. The protocol is constructed as follows.

- **Offline phase.** The trusted party $\mathcal{T}$ samples uniformly random masks $\Delta_1, \Delta_2, \gamma \leftarrow \mathbb{Z}_q$. For each $i \in \{1, 2\}$, it privately sends Δ_i to the input client $\mathcal{C}_i$. It also prepares among the computing parties (servers) $\mathcal{P}_1, \ldots, \mathcal{P}_n$ the quantum secret sharings

  $$([\![\Delta_1]\!]_1, [\![\Delta_2]\!]_2, [\![\gamma]\!]_3)$$

  within one secret-sharing batch, together with a classical n-out-of-n robust-openable sharing $\langle \gamma \rangle$.

- **Input phase.** For every $i \in \{1, 2\}$, client $\mathcal{C}_i$ broadcasts $\Delta_i' := x_i - \Delta_i$ and then retires. The parties then derive $[\![x_i]\!]_i \Leftarrow [\![\Delta_i]\!]_i + \Delta_i'$. At this point, they hold

  $$([\![x_1]\!]_1, [\![x_2]\!]_2, [\![\gamma]\!]_3).$$

- **Evaluation phase.** Using the linear homomorphism of the secret-sharing batch, the parties add $[\![x_1]\!]_1$ and $[\![x_2]\!]_2$ into $[\![\gamma]\!]_3$, and get

  $$([\![x_1]\!]_1, [\![x_2]\!]_2, [\![x_1 + x_2 + \gamma]\!]_3).$$

- **Output phase.** The parties proceed as follows.

  1. First, the parties delete $[\![x_1]\!]_1$ and $[\![x_2]\!]_2$. This includes every party deleting its shares of $[\![x_1]\!]_1$ and $[\![x_2]\!]_2$, and broadcasts the deletion certificates. Then, every other party verifies the validity of the certificates.
  2. Then, the parties reconstruct $o' = x_1 + x_2 + \gamma$ from $[\![x_1 + x_2 + \gamma]\!]_3$.
  3. Finally, the parties robustly open $\langle \gamma \rangle$ and output $o := o' - \gamma = x_1 + x_2$.

We give the following remarks on this protocol:

(1) The input masks $\{\Delta_i\}_i$ allow the trusted party to prepare the preprocessing information without knowing the clients' private inputs. For every client $\mathcal{C}_i$, the trusted party samples a uniform mask Δ_i, prepares sharings of Δ_i, and privately sends Δ_i to $\mathcal{C}_i$. The client $\mathcal{C}_i$ then broadcasts $\Delta_i' := x_i - \Delta_i$, from which the computing parties locally derive $[\![x_i]\!]_i \Leftarrow [\![\Delta_i]\!]_i + \Delta_i'$. Since Δ_i is uniform and hidden from the computing parties, the public value Δ_i' is uniform and reveals no information about x_i.

(2) Before reconstructing the designated values, the parties must delete every other sharing in the batch. This is because all sharings in the same batch use the same hidden sets of evaluation and checking positions. Reconstructing one sharing in the computational basis may reveal information about these positions, thereby compromising the deletion security of the remaining sharings in that batch. Note that *unanimously verifiable deletion* is essential here. In a privately verifiable deletion scheme, deletion certificates are verified by the trusted party $\mathcal{T}$. In our MPC protocol, however, $\mathcal{T}$ no longer participates after the offline phase, and the clients also retire after the input phase. Consequently, the deletion certificates must be verified jointly by the computing parties themselves, which requires *unanimously verifiable deletion*.

(3) The output mask γ is important for the simulation-based security proof. Our simulator for the secret-sharing batch applies only when the reconstructed value is uniformly distributed. By masking the output, the protocol replaces the reconstruction of o with the reconstruction of $o' := o + \gamma$, which is uniform over $\mathbb{Z}_q$ because γ is uniform. The task of determining the actual output is then deferred to the subsequent classical sharing: after reconstructing o', the parties robustly open the classical sharing of γ and recover $o = o' - \gamma$.

Example 2: a single multiplication. As our second example, we show how to construct a protocol computing the function

$$F_2(x_1, x_2) := x_1 \cdot x_2.$$

A natural approach is to use Beaver triples [Bea91]. We first consider the following flawed attempt.

1. In the offline phase, the trusted party $\mathcal{T}$ samples uniformly random values $\Delta_1, \Delta_2, u, v, \gamma \leftarrow \mathbb{Z}_q$, and sets $w := uv$. It privately sends Δ_i to the input client $\mathcal{C}_i$ for each $i \in \{1, 2\}$. It also prepares among the computing parties the quantum secret-sharing batch

$$([\![\Delta_1]\!]_1, [\![\Delta_2]\!]_2, [\![u]\!]_3, [\![v]\!]_4, [\![w]\!]_5, [\![\gamma]\!]_6),$$

   together with a classical n-out-of-n robust-openable sharing $\langle\gamma\rangle$.

2. In the input phase, for every $i \in \{1, 2\}$, client $\mathcal{C}_i$ broadcasts $\Delta_i' := x_i - \Delta_i$ and then retires. The parties derive $[\![x_i]\!]_i \Leftarrow [\![\Delta_i]\!]_i + \Delta_i'$. At this point, they hold

$$([\![x_1]\!]_1, [\![x_2]\!]_2, [\![u]\!]_3, [\![v]\!]_4, [\![w]\!]_5, [\![\gamma]\!]_6).$$

3. By local computation, the parties add $[\![x_1]\!]_1$ into $[\![u]\!]_3$ and $[\![x_2]\!]_2$ into $[\![v]\!]_4$, obtaining

$$([\![x_1]\!]_1, [\![x_2]\!]_2, [\![u + x_1]\!]_3, [\![v + x_2]\!]_4, [\![w]\!]_5, [\![\gamma]\!]_6).$$

4. The parties reconstruct $u' := u + x_1$ and $v' := v + x_2$ from $[\![u + x_1]\!]_3$ and $[\![v + x_2]\!]_4$. The remaining sharings are

$$([\![x_1]\!]_1, [\![x_2]\!]_2, [\![w]\!]_5, [\![\gamma]\!]_6).$$

5. The parties compute the masked output sharing
$$[\![o']\!]_6 = [\![x_1x_2 + \gamma]\!]_6 \Leftarrow [\![\gamma]\!]_6 + [\![w]\!]_5 + u'[\![x_2]\!]_2 + v'[\![x_1]\!]_1 - u'v'.$$
6. The parties delete every remaining sharing in the batch except $[\![o']\!]_6$, reconstruct $o' = x_1x_2+\gamma$, robustly open $\langle\gamma\rangle$, and output $o := o' - \gamma = x_1x_2$.

Recall from the second remark following Example 1 that reconstructing a sharing may reveal information about the common hidden evaluation and checking positions of the batch, and hence may compromise the deletion security of the remaining sharings. This creates a problem in Step 4: reconstructing $[\![u+x_1]\!]_3$ and $[\![v+x_2]\!]_4$ takes place while $[\![x_1]\!]_1$, $[\![x_2]\!]_2$, and $[\![w]\!]_5$ must still be retained for the subsequent computation. Unlike in the linear case, the parties therefore cannot simply delete all the remaining sharings before reconstruction.

Our solution is to decompose $F_2(x_1, x_2) = x_1x_2$ into two functions:
$$F_2^{(1)}(x_1, x_2, u, v) := (u + x_1, v + x_2),$$
$$F_2^{(2)}(x_1, x_2, w; u', v') := w + u'x_2 + v'x_1 - u'v'.$$
The function $F_2^{(1)}$ is linear, and once u' and v' are revealed, $F_2^{(2)}$ is also linear in its remaining secret inputs. Moreover, when $w = uv$, $u' = u + x_1$, and $v' = v + x_2$, we have
$$F_2^{(2)}(x_1, x_2, w; u', v') = x_1x_2.$$
This motivates the use of two independent secret-sharing batches. At the beginning of the online phase, they will have the form
$$\mathsf{Batch}_1 := ([\![x_1]\!]_{1,1}, [\![x_2]\!]_{1,2}, [\![u]\!]_{1,3}, [\![v]\!]_{1,4}),$$
$$\mathsf{Batch}_2 := ([\![x_1]\!]_{2,1}, [\![x_2]\!]_{2,2}, [\![w]\!]_{2,3}, [\![\gamma]\!]_{2,4}).$$
Since we now use multiple secret-sharing batches, we write $[\![s]\!]_{\kappa,\iota}$ for the ι-th sharing in the κ-th batch. Sharings within the same batch use the same hidden evaluation and checking positions, whereas the positions used by different batches are sampled independently.

To compute F_2, the parties first use Batch_1 to compute and reveal
$$(u', v') = F_2^{(1)}(x_1, x_2, u, v).$$
They then use Batch_2, together with the public values u' and v', to compute the masked output
$$o' = F_2^{(2)}(x_1, x_2, w; u', v') + \gamma = x_1x_2 + \gamma.$$
The key point is that reconstructing u' and v' from Batch_1 reveals no information about the independently sampled evaluation and checking positions of Batch_2. We now describe the resulting protocol.

- **Offline phase.** The trusted party $\mathcal{T}$ samples independent uniform values $\Delta_1, \Delta_2, u, v, \gamma \leftarrow \mathbb{Z}_q$ and sets $w := uv$. For every $i \in \{1, 2\}$, it privately sends Δ_i to the input client $\mathcal{C}_i$. It prepares among the computing parties the two quantum secret-sharing batches
$$\mathsf{Batch}_1 := ([\![\Delta_1]\!]_{1,1}, [\![\Delta_2]\!]_{1,2}, [\![u]\!]_{1,3}, [\![v]\!]_{1,4}),$$
$$\mathsf{Batch}_2 := ([\![\Delta_1]\!]_{2,1}, [\![\Delta_2]\!]_{2,2}, [\![w]\!]_{2,3}, [\![\gamma]\!]_{2,4}).$$
together with a classical n-out-of-n robust-openable sharing $\langle\gamma\rangle$.

- **Input phase.** For every $i \in \{1,2\}$, client $\mathcal{C}_i$ broadcasts $\Delta'_i := x_i - \Delta_i$ and then retires. For every $i, \kappa \in \{1,2\}$, the parties derive $[\![x_i]\!]_{\kappa,i} \Leftarrow [\![\Delta_i]\!]_{\kappa,i} + \Delta'_i$. At this point, they hold

$$\mathsf{Batch}_1 := ([\![x_1]\!]_{1,1}, [\![x_2]\!]_{1,2}, [\![u]\!]_{1,3}, [\![v]\!]_{1,4}),$$
$$\mathsf{Batch}_2 := ([\![x_1]\!]_{2,1}, [\![x_2]\!]_{2,2}, [\![w]\!]_{2,3}, [\![\gamma]\!]_{2,4}).$$

- **Evaluation phase.** The parties proceed as follows:

  1. By local computation, the parties add $[\![x_1]\!]_{1,1}$ into $[\![u]\!]_{1,3}$ and $[\![x_2]\!]_{1,2}$ into $[\![v]\!]_{1,4}$, obtaining

$$\mathsf{Batch}_1 := ([\![x_1]\!]_{1,1}, [\![x_2]\!]_{1,2}, [\![u + x_1]\!]_{1,3}, [\![v + x_2]\!]_{1,4}).$$

  2. The parties delete $[\![x_1]\!]_{1,1}$ and $[\![x_2]\!]_{1,2}$, and then reconstruct $u' := u + x_1$ and $v' := v + x_2$ from $[\![u + x_1]\!]_{1,3}$ and $[\![v + x_2]\!]_{1,4}$, respectively. After this step, every quantum sharing in Batch_1 has either been deleted or measured during reconstruction.
  3. Using the public values u' and v', the parties compute within Batch_2

$$[\![o']\!]_{2,4} = [\![x_1 x_2 + \gamma]\!]_{2,4} \Leftarrow [\![\gamma]\!]_{2,4} + [\![w]\!]_{2,3} + u'[\![x_2]\!]_{2,2} + v'[\![x_1]\!]_{2,1} - u'v'.$$

- **Output phase.** The parties delete every remaining sharing in Batch_2 except $[\![o']\!]_{2,4}$, and then reconstruct $o' = x_1 x_2 + \gamma$. They robustly open $\langle\gamma\rangle$ and output $o := o' - \gamma = x_1 x_2$.

Example 3: sequential multiplications. The idea above extends naturally to circuits with multiple multiplication gates: we can apply the above decomposition to each multiplication gate. For example, consider

$$F_3(x_1, x_2, x_3) := x_1 x_2 x_3.$$

We decompose the computation into the following three linear functions:

$$F_3^{(1)}(x_1, x_2, u_1, v_1) := (u_1 + x_1,\ v_1 + x_2),$$

$$F_3^{(2)}(x_1, x_2, x_3, w_1, u_2, v_2; u'_1, v'_1) := \big((w_1 + u'_1 x_2 + v'_1 x_1 - u'_1 v'_1) + u_2,\ x_3 + v_2\big),$$

$$F_3^{(3)}(x_1, x_2, x_3, w_1, w_2, \gamma; u'_1, v'_1, u'_2, v'_2) := \gamma + w_2 + v'_2(w_1 + u'_1 x_2 + v'_1 x_1 - u'_1 v'_1) + u'_2 x_3 - u'_2 v'_2.$$

Here, unlike in Examples 1 and 2, we include the output mask γ directly in the definition of $F_3^{(3)}$, so that $F_3^{(3)}$ already outputs the masked final value. Accordingly, we prepare the following three batches:

$$\mathsf{Batch}_1 := ([\![x_1]\!]_{1,1}, [\![x_2]\!]_{1,2}, [\![u_1]\!]_{1,3}, [\![v_1]\!]_{1,4}),$$
$$\mathsf{Batch}_2 := ([\![x_1]\!]_{2,1}, [\![x_2]\!]_{2,2}, [\![x_3]\!]_{2,3}, [\![w_1]\!]_{2,4}, [\![u_2]\!]_{2,5}, [\![v_2]\!]_{2,6}),$$
$$\mathsf{Batch}_3 := ([\![x_1]\!]_{3,1}, [\![x_2]\!]_{3,2}, [\![x_3]\!]_{3,3}, [\![w_1]\!]_{3,4}, [\![w_2]\!]_{3,5}, [\![\gamma]\!]_{3,6}),$$

where, for each $b \in \{1,2\}$, the trusted party samples $u_b, v_b \leftarrow \mathbb{Z}_q$ and sets $w_b := u_b v_b$, corresponding to the Beaver triple for the b-th multiplication gate. It also samples a uniform output mask $\gamma \leftarrow \mathbb{Z}_q$ and prepares a classical n-out-of-n robust-openable sharing $\langle\gamma\rangle$.

We first use Batch_1 to compute the masked values

$$(u'_1, v'_1) = F_3^{(1)}(x_1, x_2, u_1, v_1)$$

for the first multiplication. Next, using Batch_2 together with u'_1 and v'_1, we compute the masked values

$$(u'_2, v'_2) = F_3^{(2)}(x_1, x_2, x_3, w_1, u_2, v_2; u'_1, v'_1)$$

for the second multiplication. Finally, using Batch_3 together with u'_1, v'_1, u'_2, v'_2, we compute the masked output

$$o' = F_3^{(3)}(x_1, x_2, x_3, w_1, w_2, \gamma; u'_1, v'_1, u'_2, v'_2) = x_1x_2x_3 + \gamma.$$

The parties delete every other sharing in Batch_3 before reconstructing o'. They then robustly open $\langle\gamma\rangle$ and recover $F_3(x_1, x_2, x_3) = o' - \gamma = x_1x_2x_3$.

We omit the details of the protocol construction, as they follow the same pattern as in the previous examples.

Example 4: parallel multiplications. Moreover, one can do better when several multiplication gates can be evaluated in parallel. For example, consider

$$F_4(x_1, x_2, x_3, x_4) := x_1x_2 + x_3x_4.$$

Since the two multiplication gates are independent, we can decompose the computation into only two linear functions:

$$F_4^{(1)}(x_1, x_2, x_3, x_4, u_1, v_1, u_2, v_2) := (u_1 + x_1,\ v_1 + x_2,\ u_2 + x_3,\ v_2 + x_4),$$

$$\begin{aligned} &F_4^{(2)}(x_1, x_2, x_3, x_4, w_1, w_2, \gamma; u'_1, v'_1, u'_2, v'_2) \\ &\quad := \gamma + (w_1 + u'_1x_2 + v'_1x_1 - u'_1v'_1) + (w_2 + u'_2x_4 + v'_2x_3 - u'_2v'_2). \end{aligned}$$

Accordingly, we prepare the following two batches:

$$\begin{aligned} \mathsf{Batch}_1 &:= (\llbracket x_1 \rrbracket_{1,1}, \llbracket x_2 \rrbracket_{1,2}, \llbracket x_3 \rrbracket_{1,3}, \llbracket x_4 \rrbracket_{1,4}, \llbracket u_1 \rrbracket_{1,5}, \llbracket v_1 \rrbracket_{1,6}, \llbracket u_2 \rrbracket_{1,7}, \llbracket v_2 \rrbracket_{1,8}), \\ \mathsf{Batch}_2 &:= (\llbracket x_1 \rrbracket_{2,1}, \llbracket x_2 \rrbracket_{2,2}, \llbracket x_3 \rrbracket_{2,3}, \llbracket x_4 \rrbracket_{2,4}, \llbracket w_1 \rrbracket_{2,5}, \llbracket w_2 \rrbracket_{2,6}, \llbracket \gamma \rrbracket_{2,7}), \end{aligned}$$

where, for each $b \in \{1, 2\}$, the trusted party samples $u_b, v_b \leftarrow \mathbb{Z}_q$ and sets $w_b := u_bv_b$, corresponding to the Beaver triple for the b-th multiplication gate. It also samples a uniform output mask $\gamma \leftarrow \mathbb{Z}_q$ and prepares a classical n-out-of-n robust-openable sharing $\langle\gamma\rangle$.

We first use Batch_1 to compute the masked values

$$(u'_1, v'_1, u'_2, v'_2) = F_4^{(1)}(x_1, x_2, x_3, x_4, u_1, v_1, u_2, v_2).$$

Then, using Batch_2 together with the revealed values u'_1, v'_1, u'_2, v'_2, we compute the masked output

$$o' = F_4^{(2)}(x_1, x_2, x_3, x_4, w_1, w_2, \gamma; u'_1, v'_1, u'_2, v'_2) = x_1x_2 + x_3x_4 + \gamma.$$

Before reconstructing o', the parties delete every other sharing in Batch_2. They then robustly open $\langle\gamma\rangle$ and recover $F_4(x_1, x_2, x_3, x_4) = o' - \gamma = x_1x_2 + x_3x_4$.

Consequently, the number of additional batches required depends only on the multiplication depth of the circuit, rather than on the total number of multiplication gates. In particular, when the functionality is a public polynomial, the circuit can be chosen so that its multiplication depth is at most the degree of the polynomial.

1.3 Security Proof Ideas

In this subsection, we outline the main ideas behind the security proofs of our secret-sharing scheme and MPC protocol. For secret sharing, we adapt the general strategy of Bartusek and Raizes [BR24], with additional techniques to handle batches and unanimously verifiable deletion. Notably, we also identify and fix a bug in their original proof. For MPC, we combine the secret-sharing proof with the standard simulation framework of classical MPC, while addressing several additional subtleties that arise in our setting. We start with the proof in [BR24].

The proof strategy in [BR24]. Now we explain how Bartusek and Raizes prove the adaptive certified deletion security for their secret-sharing construction in [BR24, Section 7]. We refer to [BR24, Section 2.2] for a more detailed overview.

1. **Step 1: Lazy sampling.** At the beginning of the experiment, sample the checking positions and their values. Partition each share's evaluation points uniformly at random into two sets, referred to as the left set and the right set. Sample all left evaluation points uniformly at random.

   For the first t corruptions, sample the corresponding right evaluation points uniformly upon corruption. For the subsequent corruptions, obtain the right evaluation points by interpolation. Note that once at least t shares have been corrupted, the left evaluations of all shares and the right evaluations of the corrupted shares, together with the fixed point $f(0)$, provide sufficiently many points for interpolation.

2. **Step 2: Purification.** Let C denote the challenger's private register and A the corresponding register to be delivered. The purification uses the identity

   $$q^{-1/2} \sum_{x \in \mathbb{Z}_q} |x\rangle_{\mathsf{C}} \otimes |x\rangle_{\mathsf{A}} = q^{-1/2} \sum_{x \in \mathbb{Z}_q} \mathsf{QFT}^{-1}|x\rangle_{\mathsf{C}} \otimes \mathsf{QFT}|x\rangle_{\mathsf{A}}.$$

   For a checking position, prepare this EPR pair and measure C in the inverse-QFT basis. The outcome is recorded in the verification key, and A is left in the corresponding Fourier-basis state.

   For the left evaluation points of all shares and the right evaluation points of the first t corrupted shares, prepare these EPR pairs without measuring C. The challenger retains C and delivers A upon corruption.

   For the right evaluation points of the subsequently corrupted $n-t$ shares, coherently compute their values from the retained interpolation inputs into new C registers. Copy these values into A in the computational basis and deliver A.

3. **Step 3: Additional deletion projections.** For each deleted share, the challenger introduces an additional projective measurement that checks whether the inverse-QFT labels of its private evaluation registers C are close in Hamming distance to the corresponding deletion certificate. If not, the measurement rejects and the experiment aborts.

   Bartusek and Raizes argue that, when these additional measurements accept, the shares delivered to the adversary look like fresh random states. Concretely, for every $i \in [n-t]$, they pair the $(t+i)$-th corrupted share with the i-th deleted share. They use a seedless quantum

extractor [BR24, Theorem 3] to extract randomness from the deleted share's private registers into the right evaluation points of the corresponding corrupted share. They therefore claim that a simulator which replaces all shares by fresh random states preserves the adversary's view in the hybrid with additional projections.

An issue in the proof of [BR24]. However, their proof has an issue: the additional deletion projections can have different acceptance probabilities in the interpolating hybrid and the simulator, contrary to what their argument assumes. Consider the following counterexample. Suppose $t = 1$ and $n \geq 3$. Suppose $t = 1$ and $n \geq 3$. For each party $\mathcal{P}_i$, let C_i denote the challenger's private registers corresponding to the evaluation points of P_i's share, and let A_i denote the corresponding registers delivered to the adversary upon corruption of $\mathcal{P}_i$. Consider the following adversary $\mathcal{A}$: it corrupts $\mathcal{P}_1$, honestly deletes its share, corrupts $\mathcal{P}_2$, and honestly deletes its share.

In the simulator, the additional projections for both $\mathcal{P}_1$ and $\mathcal{P}_2$ accept with probability 1. Indeed, each share's data registers are prepared using fresh EPR pairs, so honest deletion leaves the corresponding private registers in the inverse-QFT basis state specified by the certificate. In the interpolating hybrid, the projection for $\mathcal{P}_1$ also accepts with probability 1, since its data registers are prepared in the same way. However, the same does not hold for the projection for $\mathcal{P}_2$. Since A_1 has been measured by $\mathcal{A}$ on the QFT basis, C_1 has also collapsed to the corresponding inverse-QFT basis state. If we coherently interpolate from C_1 to C_2 and copy C_2 to A_2 on the computational basis, we can no longer treat the evaluation points on $(\mathsf{C}_2, \mathsf{A}_2)$ as fresh EPR pairs, since we cannot just ignore the entanglement with the state on C_1.

Now consider the honest deletion of $\mathcal{P}_2$: the adversary measures A_2 on the QFT basis and submits cert_2. This certificate should be valid with probability 1. However, since C_1 and C_2 remain in an entangled superposition at this point, the additional projection on C_2 rejects with non-negligible probability. Thus, the acceptance probabilities of this additional projection in the interpolating hybrid and the simulator differ by a non-negligible amount.

The following four-qubit example further illustrates this discrepancy. Write $|+\rangle := (|0\rangle + |1\rangle)/\sqrt{2}$ and $|\Phi^+\rangle := (|00\rangle + |11\rangle)/\sqrt{2}$. Initially, prepare $\mathsf{C}_1\mathsf{A}_1$ in $|\Phi^+\rangle$ and initialize $\mathsf{C}_2\mathsf{A}_2$ to $|00\rangle$. Measure A_1 in the Hadamard basis. Assume the outcome is $|+\rangle$. This leaves $\mathsf{C}_1\mathsf{A}_1$ in $|+\rangle_{\mathsf{C}_1}|+\rangle_{\mathsf{A}_1}$, so the projector $\mathbf{\Pi} := |+\rangle\langle+|$ on C_1 accepts with probability 1.

Now compare two computations. The first applies a CNOT from C_1 to C_2, followed by a CNOT from C_2 to A_2. The second prepares a fresh EPR pair in $\mathsf{C}_2\mathsf{A}_2$. The resulting states are

$$|\Psi_{\mathsf{Hyb}}\rangle = |+\rangle_{\mathsf{A}_1} \frac{(|000\rangle + |111\rangle)_{\mathsf{C}_1\mathsf{C}_2\mathsf{A}_2}}{\sqrt{2}}, \ |\Psi_{\mathsf{Sim}}\rangle \ = |+\rangle_{\mathsf{C}_1}|+\rangle_{\mathsf{A}_1}|\Phi^+\rangle_{\mathsf{C}_2\mathsf{A}_2}.$$

In both cases, A_2 is maximally mixed.

Next, measure A_2 in the Hadamard basis. Assume again the outcome is $|+\rangle$, which occurs with probability $1/2$ in both cases. Starting from $|\Psi_{\mathsf{Sim}}\rangle$, this leaves C_2 in $|+\rangle$, so the projector $\mathbf{\Pi}$ on C_2 accepts with probability 1. Starting from $|\Psi_{\mathsf{Hyb}}\rangle$, it leaves $\mathsf{C}_1\mathsf{C}_2$ in $|\Phi^+\rangle$, so C_2 alone remains maximally mixed. Consequently, $\mathbf{\Pi}$ on C_2 accepts with probability only $1/2$, far below the $1 - \mathrm{negl}(\lambda)$ claimed in [BR24].

Repairing the proof. Consider again the four-qubit example above. If we replace the projector $|+\rangle\langle+|$ on C_2 in the first computation by the joint projector $|\Phi^+\rangle\langle\Phi^+|$ on $\mathsf{C}_1\mathsf{C}_2$, then this projector

will accept with probability 1, matching the accepting probability of $\mathbf{\Pi}$ on C_2 in the simulator case. Our repair similarly replaces the local deletion projections in the interpolating hybrid by global projections that also act on the retained interpolation sources.

Let $\mathbf{y} = (\mathbf{y}_i)_i$ denote the outcomes of "conceptual" QFT-basis measurements (which are not actually performed) on all delivered data registers A_i in the purified generation circuit, before adversarial processing. Let $|\mathsf{priv}(\mathbf{y})\rangle$ be the corresponding conditional state of the private registers retained by the challenger from generating the data evaluations. Let $\mathcal{G}_{\mathsf{cert}}$ be the set of joint labels $\mathbf{y}$ such that, for every deleted $\mathcal{P}_i$, $\mathbf{y}_i$ is close in Hamming distance to the entries of cert_i at that share's data positions. We define the global projector as

$$\mathbf{\Pi}^{\mathsf{glob}}_{\mathsf{cert}} := \sum_{\mathbf{y}\in\mathcal{G}_{\mathsf{cert}}} |\mathsf{priv}(\mathbf{y})\rangle\langle\mathsf{priv}(\mathbf{y})|.$$

Under the corruption bound and preceding checks, we prove that the private states indexed by all labels allowed before the current check form an orthonormal family. In the simulator, we retain the original local Fourier projector on the deleted share's private data registers.

We consider the corruption and deletion events in time order and prove the following stronger statement by induction: after each event, the hybrid and the simulator have the same reduced state on a collection of registers R containing the adversary's registers and some other registers. In particular, we associate each label $\mathbf{y}$ with an unnormalized state $|\chi_{\mathbf{y}}\rangle_{\mathsf{R}}$ and show that the corresponding states in the hybrid and the simulator are identical. Both the global projector in the hybrid and the local projector in the simulator retain exactly the labels $\mathbf{y} \in \mathcal{G}_{\mathsf{cert}}$, so they accept with the same probability. Tracing out all registers not held by the adversary gives the same adversarial view in the two experiments.

Extension to the secret sharing batch and MPC. Beyond this repair, the security proof of our secret-sharing batch requires several additional ingredients. First, in the unanimously verifiable deletion setting, corrupting a party reveals its verification information for the other parties. The lazy generation must remain consistent with every checking position and value already disclosed. Second, all sharings within the same batch reuse the same hidden checking positions. We must therefore establish the correspondence jointly across the batch, allowing arbitrary correlations between its coordinates. Third, we must handle honest deletion: when an honest party deletes its share, the challenger must generate a certificate consistent with the adversary's current view.

For MPC, the overall proof follows the standard simulation-based paradigm of classical MPC. In the offline phase, the simulator prepares the quantum batches using independent uniform secrets, without knowing the uncorrupted clients' inputs. In the input phase, it simulates the uncorrupted clients' public input messages using independent uniform values. In the evaluation phase, it simulates the honest computing parties' operations and public messages and tracks how the sharings evolve. In the output phase, it programs the classical output-mask sharings so that the final outputs open to the values provided by the ideal functionality. Throughout the execution, it answers corruption and deletion queries using the simulated states. A hybrid argument analogous to that for our secret-sharing batch then establishes that the simulated and real adversarial views are statistically indistinguishable.

Combining the secret-sharing proof with this classical MPC simulation framework gives rise to several additional issues. First and foremost, we need to reconstruct some of the sharings in each

batch. We therefore cannot simply follow the secret-sharing proof and replace all corrupted shares with random states: we must preserve the polynomial information in the sharings to be reconstructed and replace only the corrupted shares of those to be deleted. In particular, the replacement states must be random after evaluation. Second, the simulator must prepare all quantum sharings in the preprocessing phase, before the uncorrupted clients' inputs are known. To simulate these sharings without knowing the inputs, we use the fact that every value reconstructed from a quantum batch is masked by an independent uniform value. The simulator can therefore prepare the batches using independent uniform secrets and later program the classical output-mask sharings so that the final outputs open to the values provided by the ideal functionality. Finally, corrupted computing parties may send malformed reconstruction messages. We must therefore show that the consistency checks in the reconstruction procedure ensure that every reconstruction either returns the correct value or causes the protocol to abort.

1.4 Related Work

Secure computation with everlasting security transfer. Bartusek and Khurana [BK23] introduced *everlasting security transfer* (EST) and constructed two-party and multi-party secure computation satisfying this notion. More specifically, after the protocol execution, a fixed corrupted subset may comply with deletion requests by removing information about the other parties' inputs from its view, after which the resulting residual view enjoys statistical (everlasting) security. Thus, deletion serves as a post-execution security-transfer mechanism, rather than an operation that can be performed incrementally during the computation. Our goal is different. Certified deletion is integrated into the MPC execution itself: the computing parties directly evaluate on certified-deletable secret-sharing states, and a party may delete its current quantum state, have this deletion verified by the other computing parties, and then continue the computation. Moreover, corruptions occur adaptively throughout the execution, and a successfully deleted corrupted state no longer counts toward the bound on currently undeleted corrupted servers, thereby allowing further corruptions later in the execution.

Software with certified deletion. The secure software leasing techniques of Bartusek et al. [BGK$^+$24] can be viewed, to some extent, as a form of "outsourced computation with certified deletion." In the *non-interactive* setting, they propose *obfuscation* with certified deletion, assuming post-quantum iO and one-way functions. Here the protected object is the program itself:an evaluator can run the encoded program on inputs of its choice and later verifiably delete it. Our MPC addresses a different security goal: the circuit to be evaluated is public and need not be protected or deleted; instead, the sensitive information consists of the clients' private inputs. In the *interactive* setting, they propose *blind delegation* with certified deletion, assuming post-quantum hardness of LWE. The client delegates computation on a private input to a server and can request and verify deletion of the encoded input at the end of the protocol. In our MPC protocol, as a comparison, the input clients retire after submitting their inputs, and deletions are verified by the servers themselves. Individual servers may delete their current states while the remaining servers continue computing. Moreover, our protocol achieves information-theoretic security throughout the execution in the trusted-preprocessing model.

Comparison with classical MPC. Without the certified-deletion requirement, information-theoretically secure classical MPC is well understood. In particular, classical protocols achieve malicious security with an honest majority ($n > 2t$), while in the preprocessing model, security with abort can even be achieved against a dishonest majority ($n > t$, e.g. [BDOZ11]). Thus, the corruption threshold of our MPC protocol is not an improvement over classical MPC; the additional guarantee provided by our construction is the ability to verify deletion of the computing parties' states.

2 Preliminaries

Let $[n] := \{1, 2, \ldots, n\}$. For any n-dimensional vector $\mathbf{v}$ and any $i \in [n]$, let $v[i]$ denote the i-th entry of $\mathbf{v}$. For any subset $\mathcal{I} = \{i_1, i_2, \ldots, i_k\} \subseteq [n]$ with $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, let $\mathbf{v}[\mathcal{I}]$ denote the vector $(v[i_1], \ldots, v[i_k])^T$. For any collection $\mathcal{X} = \{(i, x_i)\}_{i \in \mathcal{I}}$, let $\mathcal{X}[i] := x_i$. Let $\mathbf{I} := \mathrm{diag}(1, 1, ..., 1)$ be the identity matrix. For any matrix $\mathbf{A}$, let $\mathbf{A}^T$ denote its transpose. If $\mathbf{A}$ is invertible, then $\mathbf{A}^{-1}$ denotes its inverse. Let $\mathbf{A}^{-T} := (\mathbf{A}^{-1})^T$ For any finite set $\mathcal{S}$, the notation $x \leftarrow \mathcal{S}$ means that x is sampled uniformly at random from $\mathcal{S}$.

2.1 Lemmas on Quantum Computation

In this subsection, we present some lemmas about quantum computation.

Lemma 4 (Gentle Measurement [Win99])**.** *Let ρ_{X} be a quantum state and let $(\mathbf{\Pi}, \mathbf{I} - \mathbf{\Pi})$ be a projective measurement on X such that $\mathrm{tr}[\mathbf{\Pi}\rho] \geq 1 - \delta$. Let*

$$\rho' = \frac{\mathbf{\Pi}\rho\mathbf{\Pi}}{\mathrm{tr}[\mathbf{\Pi}\rho]}$$

be the state after applying $(\mathbf{\Pi}, \mathbf{I} - \mathbf{\Pi})$ to ρ and post-selecting on obtaining the first outcome. Then $\mathsf{TD}(\rho, \rho') \leq 2\sqrt{\delta}$.

The next lemma states that for a quantum state, if its support in the Fourier basis has low Hamming weight, then applying an appropriate linear extractor yields uniform randomness, even in the presence of quantum side information.

Lemma 5 (High-Rate Seedless Quantum Extractors, Theorem 3 of [BR24])**.** *Let $\mathbb{F}$ be a finite field of order p^k. Let $\mathsf{X} := \mathsf{X}_1, \ldots, \mathsf{X}_M$ be a register containing M $\mathbb{F}$-qudits, and consider any quantum state*

$$|\gamma\rangle_{\mathsf{A},\mathsf{X}} = \sum_{\substack{\mathbf{u} \in \mathbb{F}^M \\ \mathsf{HW}(\mathbf{u}) < \frac{M-m}{2}}} |\psi_{\mathbf{u}}\rangle_{\mathsf{A}} \otimes |\mathbf{u} + \mathbf{w}\rangle_{\mathsf{X}}$$

for some integer $m \leq M$ and some fixed string $\mathbf{w} \in \mathbb{F}^M$. Let $\mathbf{R} \in \mathbb{F}^{m \times M}$ be a matrix such that every set of m columns of $\mathbf{R}$ are linearly independent.

Let $\rho_{\mathsf{A},\mathsf{Y}}$ be the mixed state that results from the following procedure:

1. *Apply a quantum Fourier transform over $\mathbb{F}$'s additive group $\mathbb{Z}_p^k$ to each register X_i. In other words, interpret X_i as a sequence of registers $\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,k}$ containing $\mathbb{Z}_p$-qudits, then apply a quantum Fourier transform mod p to each $\mathsf{X}_{i,j}$.*

2. *Initialize a fresh register* Y*, and apply the isometry* $|\mathbf{x}\rangle_{\mathsf{X}} \to |\mathbf{x}\rangle_{\mathsf{X}} \otimes |\mathbf{R}\mathbf{x}\rangle_{\mathsf{Y}}$.

3. *Trace out register* X.

Then

$$\rho_{\mathsf{A},\mathsf{Y}} = \mathrm{tr}_{\mathsf{X}}[|\gamma\rangle\langle\gamma|] \otimes \left(\frac{1}{|\mathbb{F}|^m} \sum_{y \in \mathbb{F}^m} |y\rangle\langle y| \right).$$

Lemma 6. *For any prime* q*, any* $a, b, c \in \mathbb{Z}_q$ *let* $\mathbf{U}_c|a\rangle|b\rangle := |a\rangle|ca+b\rangle$*. Then*

$$\mathbf{U}_c(\mathsf{QFT}|a\rangle \otimes \mathsf{QFT}|b\rangle) = \mathsf{QFT}|a - cb\rangle \otimes \mathsf{QFT}|b\rangle.$$

Proof. Let $w_q := e^{2\pi\sqrt{-1}/q}$. We have

$$\begin{aligned}
\mathbf{U}_c(\mathsf{QFT}|a\rangle \otimes \mathsf{QFT}|b\rangle) &= \mathbf{U}_c \sum_{x,y \in \mathbb{Z}_q} w_q^{ax+by} |x\rangle|y\rangle \\
&= \sum_{x,y \in \mathbb{Z}_q} w_q^{ax+by} |x\rangle|cx+y\rangle \\
&= \sum_{x,y \in \mathbb{Z}_q} w_q^{ax+b(y-cx)} |x\rangle|y\rangle \\
&= \sum_{x \in \mathbb{Z}_q} w_q^{(a-cb)x} |x\rangle \otimes \sum_{y \in \mathbb{Z}_q} w_q^{by} |y\rangle \\
&= \mathsf{QFT}|a - cb\rangle \otimes \mathsf{QFT}|b\rangle.
\end{aligned}$$

2.2 Polynomial Interpolation

We now recall some useful facts about polynomial interpolation over finite fields, starting with Lagrange interpolation in matrix form. The proof of the following lemma can be found, e.g., in [BR24, Section 3.3].

Lemma 7 (Lagrange Interpolation)**.** *Let* $\mathbb{F}$ *be a field and* $d \in [0, |\mathbb{F}|)$ *be an integer. For any distinct evaluation points* $x_1, \ldots, x_{d+1} \in \mathbb{F}$ *and any target points* $x'_1, \ldots, x'_s \in \mathbb{F}$*, there exists a matrix* $\mathbf{R} \in \mathbb{F}^{s \times (d+1)}$ *such that for every polynomial* $f \in \mathbb{F}[X]$ *with* $\deg(f) \leq d$*,*

$$\big(f(x'_1), \ldots, f(x'_s)\big)^{\mathsf{T}} = \mathbf{R}\big(f(x_1), \ldots, f(x_{d+1})\big)^{\mathsf{T}}.$$

Moreover, if $s \leq d+1$*, the target points* $x'_1, \ldots, x'_s$ *are pairwise distinct, and*

$$\{x'_1, \ldots, x'_s\} \cap \{x_1, \ldots, x_{d+1}\} = \emptyset,$$

then any s *columns of* $\mathbf{R}$ *are linearly independent.*

For any field $\mathbb{F}$, any integer $d \in [0, |\mathbb{F}|)$, define $\mathsf{Interpolate}$ as the function that takes in $d+1$ distinct points $x_1, \ldots, x_{d+1} \in \mathbb{F}$ with their evaluations $f(x_1), \ldots, f(x_{d+1})$, as well as several target points $x'_1, \ldots, x'_s \in \mathbb{F}$, and outputs $f(x'_1), \ldots, f(x'_s)$.

Polynomial interpolation also underpins error correction, as captured by the following lemma on Reed-Solomon decoding.

Lemma 8 (Safe Reed-Solomon Reconstruction). *Let $\mathbb{F}$ be a finite field, and let $f \in \mathbb{F}[X]$ be a polynomial of degree at most d. Suppose we are given s distinct evaluation points $x_1, \ldots, x_s \in \mathbb{F}$ together with corresponding values $y_1, \ldots, y_s \in \mathbb{F}$, where exactly e of the pairs (x_i, y_i) satisfy $y_i \neq f(x_i)$. Then there exists a deterministic efficient algorithm, denoted by $\mathsf{Correct}_{\mathbb{F},d}$, which on any input outputs either a polynomial of degree at most d or $\perp$, and outputs f whenever $2e < s - d$.*

2.3 Hoeffding's Inequality for Sampling without Replacement

Lemma 9 (Hoeffding's Inequality for Sampling without Replacement). *Let $X = (x_1, \ldots, x_N)$ be a finite population of N points and $X_1, \ldots, X_n$ be a random sample drawn without replacement from X. Let*

$$a = \min_{1 \leq i \leq N} x_i \quad \text{and} \quad b = \max_{1 \leq i \leq N} x_i.$$

Then, for all $\varepsilon > 0$,

$$\Pr\left[\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| \geq \varepsilon\right] \leq 2\exp\left(-\frac{2n\varepsilon^2}{(b-a)^2}\right),$$

where $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ is the mean of X.

2.4 Threshold Secret Sharing with Certified Deletion

Definition 10 ([BR24], Threshold Secret Sharing with Certified Deletion). *An $(n, t, \mathbb{F})$-threshold secret sharing scheme with certified deletion is specified by an integer $n \geq 2$ indicating the number of parties, an integer $t \in [1, n-1]$ indicating the threshold, a field $\mathbb{F}$, and consists of the following algorithms:*

- $\mathsf{Split}(1^\lambda, s \in \mathbb{F})$*: This function takes in a security parameter λ, and a secret s, and outputs n share registers $\{\mathsf{sh}_i\}_{i\in[n]}$ and a classical verification key vk. Here for every $i \in [n]$, sh_i denotes party $\mathcal{P}_i$'s share.*
- $\mathsf{Reconstruct}(\mathcal{IP} \subseteq [n], \{\mathsf{sh}_i\}_{i\in\mathcal{IP}})$*: This function takes in a set of indices $\mathcal{IP}$, and a set of corresponding shares $\{\mathsf{sh}_i\}_{i\in\mathcal{IP}}$, and outputs either s or $\perp$.*
- $\mathsf{Delete}(i \in [n], \mathsf{sh}_i)$*: This function takes in an index i and a corresponding share sh_i, and outputs a certificate of deletion cert.*
- $\mathsf{Verify}(\mathsf{vk}, i, \mathsf{cert}_i)$*: This function takes in the verification key vk, an index i, and a certificate of deletion cert_i, and outputs either $\top$ (indicating accept) or $\perp$ (indicating reject).*

Remark 11. *Since this work focuses solely on threshold secret sharing, we usually omit the word "threshold" and simply refer to it as secret sharing.*

Definition 12 (Correctness of Secret Sharing with Certified Deletion). *Let $\Pi = (\mathsf{Split}, \mathsf{Reconstruct}, \mathsf{Delete}, \mathsf{Verify})$ be an $(n, t, \mathbb{F})$-threshold secret-sharing scheme with certified deletion. We say that Π is correct if the following two properties hold:*

- ***Reconstruction Correctness.*** *For all $\lambda \in \mathbb{N}$, all sets $\mathcal{IP} \subseteq [n]$ such that $|\mathcal{IP}| \geq t+1$, and all $s \in \mathbb{F}$,*

$$\Pr\left[\mathsf{Reconstruct}(\mathcal{IP} \subseteq [n], \{\mathsf{sh}_i\}_{i\in\mathcal{IP}}) = s : (\mathsf{sh}_1, \ldots, \mathsf{sh}_n, \mathsf{vk}) \leftarrow \mathsf{Split}(1^\lambda, s)\right] = 1.$$

- ***Deletion Correctness.*** *For all $\lambda \in \mathbb{N}$, all $i \in [n]$, and all $s \in \mathbb{F}$,*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, i, \mathsf{cert}_i) = \top : (\mathsf{sh}_1, \ldots, \mathsf{sh}_n, \mathsf{vk}) \leftarrow \mathsf{Split}(1^\lambda, s), \mathsf{cert}_i \leftarrow \mathsf{Delete}(i, \mathsf{sh}_i)\right] = 1.$$

Definition 13 (Adaptive Certified Deletion for Secret Sharing)**.** *Let $\Pi = (\mathsf{Split}, \mathsf{Reconstruct}, \mathsf{Delete}, \mathsf{Verify})$ be an $(n, t, \mathbb{F})$-threshold secret-sharing scheme with certified deletion. Let $\mathcal{A}$ be an adversary with internal register R initialized to a state $|\phi\rangle$, and let $s \in \mathbb{F}$ be a secret. Define the experiment $\mathsf{Exp}^{\mathsf{ss\text{-}cd}}_{\Pi,\lambda,|\phi\rangle,\mathcal{A}}(s)$ as follows:*

1. *Sample $(\mathsf{sh}_1, \ldots, \mathsf{sh}_n, \mathsf{vk}) \leftarrow \mathsf{Split}(1^\lambda, s)$. Initialize the corruption set $\mathcal{CS} \leftarrow \emptyset$ and the deletion set $\mathcal{DS} \leftarrow \emptyset$.*
2. *In each round, the adversary $\mathcal{A}$ may do one of three things:*
   (a) *End the experiment by outputting its view.*
   (b) *Delete a share by outputting an index $i \notin \mathcal{DS}$. When $\mathcal{A}$ chooses this option:*
      - *If $i \in \mathcal{CS}$, then $\mathcal{A}$ outputs a certificate cert_i. If $\mathsf{Verify}(\mathsf{vk}, i, \mathsf{cert}_i) = \top$, add i to $\mathcal{DS}$; otherwise, immediately abort the experiment and output $\perp$.*
      - *If $i \notin \mathcal{CS}$, let $\mathsf{cert}_i \leftarrow \mathsf{Delete}(i, \mathsf{sh}_i)$ and add i to $\mathcal{DS}$.*
   (c) *Corrupt a new share by outputting an index $i \notin \mathcal{CS}$. When $\mathcal{A}$ chooses this option, let $\mathcal{CS} \leftarrow \mathcal{CS} \cup \{i\}$. If $|\mathcal{CS} \backslash \mathcal{DS}| > t$, immediately abort the experiment and output $\perp$; otherwise, if $i \notin \mathcal{DS}$, give sh_i to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs its view, unless the experiment has already aborted.*

We say that Π has adaptive certified deletion security if, for any unbounded adversary $\mathcal{A}$, any state $|\phi\rangle$, and any pair of secrets $s^{(0)}, s^{(1)} \in \mathbb{F}$,

$$\mathsf{TD}\left(\mathsf{Exp}^{\mathsf{ss\text{-}cd}}_{\Pi,\lambda,|\phi\rangle,\mathcal{A}}(s^{(0)}), \mathsf{Exp}^{\mathsf{ss\text{-}cd}}_{\Pi,\lambda,|\phi\rangle,\mathcal{A}}(s^{(1)})\right) = \mathrm{negl}(\lambda).$$

2.5 Classical Secret Sharing with Robust Opening

We use a classical n-out-of-n secret-sharing protocol with robust opening for the classical sharings generated during the trusted preprocessing phase.

Definition 14 (Adaptively Programmable Secret Sharing with Robust Opening)**.** *Let $\mathbb{F}$ be a finite field. A classical secret-sharing protocol with robust opening is a pair of algorithms $\Pi_{\mathsf{ROpenSS}} = (\mathsf{Split}, \mathsf{Open})$ defined as follows.*

- $\mathsf{Split}(1^\lambda, s \in \mathbb{F})$*: The trusted preprocessing party generates an n-out-of-n sharing $\langle s \rangle = (\mathsf{sh}_1, \ldots, \mathsf{sh}_n)$ and privately sends sh_i to $\mathcal{P}_i$.*

- $\mathsf{Open}(\langle s \rangle)$: *The parties interact to open* $\langle s \rangle$*. Every honest party outputs either the shared value* s *or* $\perp$*.*

We say that Π_{ROpenSS} *is adaptively programmable and robustly openable if, against every malicious adaptive adversary corrupting at most* $n-1$ *parties, it satisfies the following properties.*

- ***Correctness.*** *For every* $\lambda \in \mathbb{N}$ *and every* $s \in \mathbb{F}$*, if* $\langle s \rangle \leftarrow \mathsf{Split}(1^\lambda, s)$ *and all parties honestly execute* $\mathsf{Open}(\langle s \rangle)$*, then every party outputs* s *with probability* 1*.*
- ***Privacy.*** *Before opening, the adversarial view associated with any corrupted set* $\mathcal{C} \subsetneq [n]$ *is statistically independent of* s*.*
- ***Statistical robust opening.*** *Except with probability at most* $\varepsilon_{\mathsf{ro}}(\lambda)$*, every non-aborting execution of* Open *returns the originally shared value, and all honest parties obtain the same result. The adversary may learn the opened value and subsequently cause all honest parties to output* $\perp$*. We require* $\varepsilon_{\mathsf{ro}}(\lambda) = \mathrm{negl}(\lambda)$*.*
- ***Statistical programmable completion.*** *For any adversarial view exposed through adaptive corruptions before the first message of* Open *is generated, and for any target value* $s^\star \in \mathbb{F}$ *specified at that point, as long as at least one party remains uncorrupted, there exists an efficient simulator that completes the unexposed honest state so that the sharing opens to* $s^\star$*. The resulting completed state and opening execution are statistically indistinguishable from an honest execution sharing* $s^\star$*, conditioned on the previously exposed view, with statistical distance at most* $\varepsilon_{\mathsf{pc}}(\lambda) = \mathrm{negl}(\lambda)$*.*

For a sharing $\langle s \rangle$, we write $s \leftarrow \mathsf{Open}(\langle s \rangle)$ for a successful opening and $\perp \leftarrow \mathsf{Open}(\langle s \rangle)$ for an aborted opening. Parallel opening of $\{\langle s_j \rangle\}_{j \in \mathcal{J}}$ is denoted by $\{s_j\}_{j \in \mathcal{J}} \leftarrow \mathsf{Open}(\{\langle s_j \rangle\}_{j \in \mathcal{J}})$; the protocol aborts if any individual opening returns $\perp$.

Information-Theoretic Realization. In the trusted-preprocessing model, such a protocol can be realized information-theoretically using BDOZ-style authenticated additive sharings [BDOZ11]. To split $s \in \mathbb{F}$, the trusted preprocessing party samples $s_1, \ldots, s_n \in \mathbb{F}$ subject to $s = \sum_{i \in [n]} s_i$. Independently for every invocation of Split and every ordered pair of distinct parties $(j, i) \in [n]^2$ with $j \neq i$, the trusted preprocessing party samples a fresh one-time information-theoretic MAC key $K_{j,i} = (\alpha_{j,i}, \beta_{j,i}) \leftarrow \mathbb{F}^2$ and sets $m_{j,i} := \alpha_{j,i} s_i + \beta_{j,i}$. Party $\mathcal{P}_i$ receives s_i and $\{m_{j,i}\}_{j \in [n] \setminus \{i\}}$, while $\mathcal{P}_j$ receives $\{K_{j,i}\}_{i \in [n] \setminus \{j\}}$.

To open the sharing, the parties proceed in two broadcast rounds.

1. Every $\mathcal{P}_i$ broadcasts s_i together with $\{m_{j,i}\}_{j \in [n] \setminus \{i\}}$.
2. Every $\mathcal{P}_j$ verifies $m_{j,i} = \alpha_{j,i} s_i + \beta_{j,i}$ for every $i \in [n] \setminus \{j\}$ and broadcasts either $\top$ or $\perp$.

If any party broadcasts $\perp$, or if any required message is missing or malformed, every honest party outputs $\perp$. Otherwise, every honest party outputs $\sum_{i \in [n]} s_i$.

For programmable completion, the simulator samples the unexposed honest shares subject to $\sum_{i \in [n]} s_i = s^\star$ and completes the remaining tags and keys consistently with the exposed view. If a tag

is exposed but its key is hidden, the key is sampled conditioned on the corresponding MAC equation; if the key is exposed but the tag is hidden, the tag is computed after fixing the share. Hence, completion has the correct conditional distribution. Over polynomially many opening attempts, the statistical MAC-forgery error is at most $\mathrm{poly}(\lambda)/|\mathbb{F}|$, which is negligible when $|\mathbb{F}| = 2^{\Omega(\lambda)}$. This claim concerns the information-theoretic authenticated-sharing layer with trusted preprocessing; the concrete preprocessing protocol of [BDOZ11] additionally uses computational assumptions.

3 Definitions of Secret-Sharing Batch and MPC with Unanimously Verifiable Deletion

3.1 Linear Homomorphic Secret-Sharing Batch with Unanimously Verifiable Deletion

In this section, we present the definitions of (threshold) secret-sharing batch with unanimously verifiable deletion.

Definition 15 (Linear Homomorphic Secret-Sharing Batch with Unanimously Verifiable Deletion)**.** *An $(n, t, M, \mathbb{F})$-linear homomorphic secret-sharing batch scheme with unanimously verifiable deletion is specified by an integer $n \geq 2$ indicating the number of parties, an integer $t \in [1, n-1]$ indicating the threshold, a positive integer M indicating the number of sharings in a batch, a field $\mathbb{F}$, and consists of the following algorithms.*

- $\mathsf{Setup}(1^\lambda)$*: This function takes in the security parameter λ, and generates n^2 master verification keys $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}$. Here for every $i_1, i_2 \in [n]$, mvk_{i_1,i_2} is $\mathcal{P}_{i_1}$'s master verification key for $\mathcal{P}_{i_2}$'s shares.*
- $\mathsf{Split}(\iota \in [M], s_\iota \in \mathbb{F}, \{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]})$*: This function takes in a secret index ι, a secret s_ι, and the master verification keys $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}$, and outputs n share registers $\{\mathsf{sh}_{i,\iota}\}_{i\in[n]}$, as well as n^2 classical verification keys $\{\mathsf{vk}_{i_1,i_2,\iota}\}_{i_1,i_2\in[n]}$ for unanimously verifiable deletion of the shares. Here for every $i \in [n]$, $\mathsf{sh}_{i,\iota}$ is $\mathcal{P}_i$'s share for s_ι, and for every $i_1, i_2 \in [n]$, $\mathsf{vk}_{i_1,i_2,\iota}$ is $\mathcal{P}_{i_1}$'s verification key for $\mathcal{P}_{i_2}$'s share of s_ι.*
- $\mathsf{AddSh}(c \in \mathbb{F}, i \in [n], \iota \in [M], \mathsf{sh}_{i,\iota})$*: This function takes in a constant c, a party index i, a share index ι, and the corresponding share $\mathsf{sh}_{i,\iota}$, and outputs a new share $\widetilde{\mathsf{sh}}_{i,\iota}$.*
- $\mathsf{ScaleAddSh}(c \in \mathbb{F}, i \in [n], \iota_1 \in [M], \iota_2 \in [M], \mathsf{sh}_{i,\iota_1}, \mathsf{sh}_{i,\iota_2})$*: This function takes in a constant c, a party index i, two share indices $\iota_1 \neq \iota_2$, and two corresponding shares $\mathsf{sh}_{i,\iota_1}, \mathsf{sh}_{i,\iota_2}$, and outputs two new shares $\widetilde{\mathsf{sh}}_{i,\iota_1}, \widetilde{\mathsf{sh}}_{i,\iota_2}$.*
- $\mathsf{ScaleAddVk}(c \in \mathbb{F}, i_1 \in [n], i_2 \in [n], \iota_1 \in [M], \iota_2 \in [M], \mathsf{mvk}_{i_1,i_2}, \mathsf{vk}_{i_1,i_2,\iota_1}, \mathsf{vk}_{i_1,i_2,\iota_2})$*: This function takes in a constant c, two party indices i_1, i_2, two verification key indices $\iota_1 \neq \iota_2$, a master verification key mvk_{i_1,i_2}, and two verification keys $\mathsf{vk}_{i_1,i_2,\iota_1}, \mathsf{vk}_{i_1,i_2,\iota_2}$, and outputs two new verification keys $\widetilde{\mathsf{vk}}_{i_1,i_2,\iota_1}, \widetilde{\mathsf{vk}}_{i_1,i_2,\iota_2}$.*
- $\mathsf{Reconstruct}(\mathcal{IP} \subseteq [n], \mathcal{IS} \subseteq [M], \{\mathsf{sh}_{i,\iota}\}_{i\in\mathcal{IP},\iota\in\mathcal{IS}})$*: This function takes in a set of party indices $\mathcal{IP} \subseteq [n]$, a set of share indices $\mathcal{IS} \subseteq [M]$, and the corresponding shares $\{\mathsf{sh}_{i,\iota}\}_{i\in\mathcal{IP},\iota\in\mathcal{IS}}$, and outputs a set of secrets $\{s_\iota\}_{\iota\in\mathcal{IS}}$, or $\perp$ to indicate that the reconstruction fails.*

- $\mathsf{Delete}(i \in [n], \iota \in [M], \mathsf{sh}_{i,\iota})$*: This function takes in a party index* i*, a share index* ι*, and the corresponding share* $\mathsf{sh}_{i,\iota}$*, and outputs a deletion certificate* $\mathsf{cert}_{i,\iota}$*.*
- $\mathsf{Verify}(i_1 \in [n], i_2 \in [n], \iota \in [M], \mathsf{mvk}_{i_1,i_2}, \mathsf{vk}_{i_1,i_2,\iota}, \mathsf{cert}_{i_2,\iota})$*: This function takes in two party indices* i_1, i_2*, a master verification key* mvk_{i_1,i_2}*, a verification key* $\mathsf{vk}_{i_1,i_2,\iota}$*, and a deletion certificate* $\mathsf{cert}_{i_2,\iota}$*, and outputs either* $\top$ *(indicating accept) or* $\bot$ *(indicating reject).*

Definition 16 (Correctness of Linear Homomorphic Secret-Sharing Batch with Unanimously Verifiable Deletion)**.** *Let* $\Pi = (\mathsf{Setup}, \mathsf{Split}, \mathsf{AddSh}, \mathsf{ScaleAddSh}, \mathsf{ScaleAddVk}, \mathsf{Reconstruct}, \mathsf{Delete}, \mathsf{Verify})$ *be an* $(n, t, M, \mathbb{F})$*-linear homomorphic secret-sharing batch scheme with unanimously verifiable deletion.*

For $\lambda \in \mathbb{N}$*, let* $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]} \leftarrow \mathsf{Setup}(1^\lambda)$*. A batch state with respect to these master verification keys is of the form*

$$\mathsf{St} = \left(\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}, \{\mathsf{vk}_{i_1,i_2,\iota}\}_{i_1,i_2\in[n],\iota\in[M]}, \{\mathsf{sh}_{i,\iota}\}_{i\in[n],\iota\in[M]}\right).$$

For $\mathbf{s} = (s_1, \ldots, s_M) \in \mathbb{F}^M$*, we write* $\mathsf{Reach}(\mathsf{St}, \mathbf{s})$ *if there exist choices of the random coins of* Split *such that, for every* $\iota \in [M]$*,*

$$(\{\mathsf{vk}_{i_1,i_2,\iota}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}_{i,\iota}\}_{i\in[n]}) = \mathsf{Split}(\iota, s_\iota, \{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]})$$

when Split *is executed using those random coins.*

We say that Π *is correct if, for every* $\lambda \in \mathbb{N}$ *and every choice of master verification keys generated as above, the following properties hold.*

- ***(Reconstruction Correctness)*** *For any batch state* St*, any* $\mathbf{s} = (s_1, \ldots, s_M) \in \mathbb{F}^M$ *such that* $\mathsf{Reach}(\mathsf{St}, \mathbf{s})$*, any* $\mathcal{IP} \subseteq [n]$ *satisfying* $|\mathcal{IP}| > t$*, and any* $\mathcal{IS} \subseteq [M]$*,*

$$\mathsf{Reconstruct}(\mathcal{IP}, \mathcal{IS}, \{\mathsf{sh}_{i,\iota}\}_{i\in\mathcal{IP},\iota\in\mathcal{IS}}) = \{s_\iota\}_{\iota\in\mathcal{IS}}.$$

- ***(Deletion Correctness)*** *For any batch state* St*, any* $\mathbf{s} \in \mathbb{F}^M$ *such that* $\mathsf{Reach}(\mathsf{St}, \mathbf{s})$*, any* $i_1, i_2 \in [n]$*, and any* $\iota \in [M]$*, let*

$$\mathsf{cert}_{i_2,\iota} \leftarrow \mathsf{Delete}(i_2, \iota, \mathsf{sh}_{i_2,\iota}).$$

  Then

$$\mathsf{Verify}(i_1, i_2, \iota, \mathsf{mvk}_{i_1,i_2}, \mathsf{vk}_{i_1,i_2,\iota}, \mathsf{cert}_{i_2,\iota}) = \top.$$

- ***(Linear Homomorphism)*** *For any batch state* St*, any* $\mathbf{s} = (s_1, \ldots, s_M) \in \mathbb{F}^M$ *such that* $\mathsf{Reach}(\mathsf{St}, \mathbf{s})$*, the following properties hold for every* $c \in \mathbb{F}$*.*

  1. *For any* $\iota \in [M]$*, let* $\widetilde{\mathsf{sh}}_{i,\iota} \leftarrow \mathsf{AddSh}(c, i, \iota, \mathsf{sh}_{i,\iota})$ *for every* $i \in [n]$*, and let* $\widetilde{\mathsf{St}}$ *be obtained from* St *by replacing* $\mathsf{sh}_{i,\iota}$ *with* $\widetilde{\mathsf{sh}}_{i,\iota}$ *for every* $i \in [n]$*. Define* $\widetilde{\mathbf{s}} = (\widetilde{s}_1, \ldots, \widetilde{s}_M)$ *by* $\widetilde{s}_\iota = s_\iota + c$ *and* $\widetilde{s}_{\iota'} = s_{\iota'}$ *for every* $\iota' \in [M] \setminus \{\iota\}$*. Then*

$$\mathsf{Reach}(\widetilde{\mathsf{St}}, \widetilde{\mathbf{s}}).$$

2. *For any distinct $\iota_1, \iota_2 \in [M]$, let*

$$(\widetilde{\mathsf{sh}}_{i,\iota_1}, \widetilde{\mathsf{sh}}_{i,\iota_2}) \leftarrow \mathsf{ScaleAddSh}(c, i, \iota_1, \iota_2, \mathsf{sh}_{i,\iota_1}, \mathsf{sh}_{i,\iota_2})$$

for every $i \in [n]$, and let

$$(\widetilde{\mathsf{vk}}_{i_1,i_2,\iota_1}, \widetilde{\mathsf{vk}}_{i_1,i_2,\iota_2}) \leftarrow \mathsf{ScaleAddVk}(c, i_1, i_2, \iota_1, \iota_2, \mathsf{mvk}_{i_1,i_2}, \mathsf{vk}_{i_1,i_2,\iota_1}, \mathsf{vk}_{i_1,i_2,\iota_2})$$

for every $i_1, i_2 \in [n]$. Let $\widetilde{\mathsf{St}}$ be obtained from St by replacing $\mathsf{sh}_{i,\iota_1}, \mathsf{sh}_{i,\iota_2}$ by $\widetilde{\mathsf{sh}}_{i,\iota_1}, \widetilde{\mathsf{sh}}_{i,\iota_2}$ for every $i \in [n]$, and $\mathsf{vk}_{i_1,i_2,\iota_1}, \mathsf{vk}_{i_1,i_2,\iota_2}$ by $\widetilde{\mathsf{vk}}_{i_1,i_2,\iota_1}, \widetilde{\mathsf{vk}}_{i_1,i_2,\iota_2}$ for every $i_1, i_2 \in [n]$. Define $\widetilde{\mathbf{s}} = (\widetilde{s}_1, \ldots, \widetilde{s}_M)$ by $\widetilde{s}_{\iota_2} = cs_{\iota_1} + s_{\iota_2}$, and $\widetilde{s}_{\iota'} = s_{\iota'}$ for every $\iota' \in [M] \setminus \{\iota_2\}$. Then

$$\mathsf{Reach}(\widetilde{\mathsf{St}}, \widetilde{\mathbf{s}}).$$

Definition 17 (Adaptive Unanimously Verifiable Deletion Security of Secret-Sharing Batch)**.** *Let $\Pi = (\mathsf{Setup}, \mathsf{Split}, \mathsf{AddSh}, \mathsf{ScaleAddSh}, \mathsf{ScaleAddVk}, \mathsf{Reconstruct}, \mathsf{Delete}, \mathsf{Verify})$ be an $(n, t, M, \mathbb{F})$-linearly homomorphic secret-sharing batch scheme with unanimously verifiable deletion. Let $\mathcal{A}$ be an adversary with internal register R which is initialized to a state $|\phi\rangle$, and let $\{s_\iota \in \mathbb{F}\}_{\iota \in [M]}$ be a set of secrets. Define the experiment $\mathsf{Exp}^{\mathsf{ss\text{-}uvd}}_{\Pi,\lambda,|\phi\rangle,\mathcal{A}}(\{s_\iota\}_{\iota\in[M]})$ as follows:*

- *Let $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]} \leftarrow \mathsf{Setup}(1^\lambda)$, and for all $\iota \in [M]$,*

$$(\{\mathsf{vk}_{i_1,i_2,\iota}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}_{i,\iota}\}_{i\in[n]}) \leftarrow \mathsf{Split}(\iota, s_\iota, \{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}).$$

  Initialize the corruption set $\mathcal{CS} \leftarrow \emptyset$ and the deletion set $\mathcal{DS} \leftarrow \emptyset$.

- *In each round, the adversary $\mathcal{A}$ may do one of three things:*
  - *(a) End the experiment by outputting its view.*
  - *(b) Delete a party's data by outputting an index $i_1 \notin \mathcal{DS}$. When $\mathcal{A}$ chooses this option:*
    - *If $i_1 \in \mathcal{CS}$, then $\mathcal{A}$ outputs a set of certificates $\{\mathsf{cert}_{i_1,\iota}\}_{\iota\in[M]}$. Then, the challenger $\mathcal{C}$ checks whether $\mathsf{Verify}(i_2, i_1, \iota, \mathsf{mvk}_{i_2,i_1}, \mathsf{vk}_{i_2,i_1,\iota}, \mathsf{cert}_{i_1,\iota}) = \top$ for every $\iota \in [M]$ and $i_2 \notin \mathcal{CS}$. If so, add i_1 to $\mathcal{DS}$; otherwise, immediately abort the experiment and output $\bot$.*
    - *If $i_1 \notin \mathcal{CS}$, for every $\iota \in [M]$, let $\mathsf{cert}_{i_1,\iota} \leftarrow \mathsf{Delete}(i_1, \iota, \mathsf{sh}_{i_1,\iota})$. The challenger $\mathcal{C}$ gives $\mathcal{A}$ the set of certificates $\{\mathsf{cert}_{i_1,\iota}\}_{\iota\in[M]}$. Add i_1 to $\mathcal{DS}$.*
  - *(c) Corrupt a new party's data by outputting an index $i_1 \notin \mathcal{CS}$. When $\mathcal{A}$ chooses this option, let $\mathcal{CS} \leftarrow \mathcal{CS} \cup \{i_1\}$. If $|\mathcal{CS}\setminus\mathcal{DS}| > t$, immediately abort the experiment and output $\bot$; otherwise,*
    - *If $i_1 \notin \mathcal{DS}$, give $(\{\mathsf{mvk}_{i_1,i_2}\}_{i_2\in[n]}, \{\mathsf{vk}_{i_1,i_2,\iota}\}_{i_2\in[n],\iota\in[M]}, \{\mathsf{sh}_{i_1,\iota}\}_{\iota\in[M]})$ to $\mathcal{A}$.*
    - *If $i_1 \in \mathcal{DS}$, give $(\{\mathsf{mvk}_{i_1,i_2}\}_{i_2\in[n]}, \{\mathsf{vk}_{i_1,i_2,\iota}\}_{i_2\in[n],\iota\in[M]})$ together with all residual classical information left in $\mathcal{P}_{i_1}$'s local state after deletion to $\mathcal{A}$.*
- *$\mathcal{A}$ outputs its view, unless the experiment has already aborted.*

We say that Π has adaptive unanimously verifiable deletion security if, for any unbounded adversary $\mathcal{A}$, any state $|\phi\rangle$, and any pair of secret batches $\{s_\iota^{(0)}\}_{\iota\in[M]}$ and $\{s_\iota^{(1)}\}_{\iota\in[M]}$,

$$\mathsf{TD}\left(\mathsf{Exp}^{\mathsf{ss\text{-}uvd}}_{\Pi,\lambda,|\phi\rangle,\mathcal{A}}(\{s_\iota^{(0)}\}_{\iota\in[M]}), \mathsf{Exp}^{\mathsf{ss\text{-}uvd}}_{\Pi,\lambda,|\phi\rangle,\mathcal{A}}(\{s_\iota^{(1)}\}_{\iota\in[M]})\right) = \mathrm{negl}(\lambda).$$

Remark 18. *Compared to the adaptive certified deletion security, we specify the scenario where an honest party deletes its share. This is necessary because in the case of unanimous verification, even though a party has deleted its share, the adversary can still corrupt its verification keys. As a comparison, in the case of certified deletion, each party only possesses its share.*

In addition, we define unanimously verifiable deletion security only for freshly generated sharings. This suffices for our application because we invoke the present secret-sharing security definition only for corruptions and deletions of freshly generated sharings. Although in the MPC protocol, deletion may also be performed on evaluated sharings, the security relies on arguments different from those captured by the present secret-sharing security definition. We therefore defer its treatment to the security analysis of the MPC protocol. More generally, one could formulate the definition over any batch state St *and secret vector* $\mathbf{s}$ *satisfying* $\mathsf{Reach}(\mathsf{St}, \mathbf{s})$*, thereby covering sharings obtained through a sequence of linear evaluations. We restrict attention to freshly generated sharings here for simplicity.*

The following theorem shows that unanimously verifiable deletion is a stronger notion than certified deletion.

Theorem 19. *Let* $n \geq 2$ *be an integer, let* $t \in [1, n-1]$*, and let* $\mathbb{F}$ *be a field. If there exists an* $(n, t, 1, \mathbb{F})$*-secret-sharing batch scheme with adaptive unanimously verifiable deletion security, then there exists an* $(n, t, \mathbb{F})$*-secret-sharing scheme with adaptive certified deletion security.*

Proof. Suppose the $(n, t, 1, \mathbb{F})$-secret-sharing batch scheme with adaptive unanimously verifiable deletion security is

$$\Pi := (\mathsf{Setup}, \mathsf{Split}, \mathsf{AddSh}, \mathsf{ScaleAddSh}, \mathsf{ScaleAddVk}, \mathsf{Reconstruct}, \mathsf{Delete}, \mathsf{Verify}).$$

We construct an $(n, t, \mathbb{F})$-secret-sharing scheme $\Pi' := (\mathsf{Split}', \mathsf{Reconstruct}', \mathsf{Delete}', \mathsf{Verify}')$ with adaptive certified deletion security as follows.

- $\mathsf{Split}'(1^\lambda, s \in \mathbb{F})$: Sample $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]} \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$, and run

  $$(\{\mathsf{vk}_{i_1,i_2,1}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}_{i,1}\}_{i\in[n]}) \leftarrow \Pi.\mathsf{Split}(1, s, \{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}).$$

  For every $i \in [n]$, set $\mathsf{sh}'_i := \mathsf{sh}_{i,1}$, and set $\mathsf{vk}' := (\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}, \{\mathsf{vk}_{i_1,i_2,1}\}_{i_1,i_2\in[n]})$. Output $(\{\mathsf{sh}'_i\}_{i\in[n]}, \mathsf{vk}')$.
- $\mathsf{Reconstruct}'(\mathcal{IP} \subseteq [n], \{\mathsf{sh}'_i\}_{i\in\mathcal{IP}})$: Run $\mathsf{out} \leftarrow \Pi.\mathsf{Reconstruct}(\mathcal{IP}, \{1\}, \{\mathsf{sh}'_i\}_{i\in\mathcal{IP}})$. If $\mathsf{out} = \perp$, output $\perp$; otherwise, parse $\mathsf{out} = \{s\}$ and output s.
- $\mathsf{Delete}'(i \in [n], \mathsf{sh}'_i)$: Output $\mathsf{cert}'_i \leftarrow \Pi.\mathsf{Delete}(i, 1, \mathsf{sh}'_i)$.
- $\mathsf{Verify}'(\mathsf{vk}', i \in [n], \mathsf{cert}'_i)$: Parse $\mathsf{vk}' = (\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}, \{\mathsf{vk}_{i_1,i_2,1}\}_{i_1,i_2\in[n]})$. Output $\top$ if $\Pi.\mathsf{Verify}(j, i, 1, \mathsf{mvk}_{j,i}, \mathsf{vk}_{j,i,1}, \mathsf{cert}'_i) = \top$ for every $j \in [n]$, and output $\perp$ otherwise.

The reconstruction correctness of Π' follows immediately from the reconstruction correctness of Π. Similarly, the deletion correctness of Π' follows from the unanimous deletion correctness of Π, since an honestly generated deletion certificate is accepted by every verifier $j \in [n]$.

It remains to prove adaptive certified deletion security. Suppose there exists an adversary $\mathcal{A}'$ that distinguishes the adaptive certified deletion experiments of Π' for two secrets $s^{(0)}, s^{(1)} \in \mathbb{F}$ with

non-negligible advantage. We construct an adversary $\mathcal{A}$ against the adaptive unanimously verifiable deletion security of Π for the singleton batches $\{s^{(0)}\}$ and $\{s^{(1)}\}$.

The adversary $\mathcal{A}$ internally runs $\mathcal{A}'$ and forwards each corruption query to the unanimously verifiable deletion experiment. When P_i is corrupted, $\mathcal{A}$ keeps $\{\mathsf{mvk}_{i,j}\}_{j\in[n]}$ and $\{\mathsf{vk}_{i,j,1}\}_{j\in[n]}$ obtained from the challenger, and forwards $\mathsf{sh}_{i,1}$ to $\mathcal{A}'$ if $i \notin \mathcal{DS}$. For a deletion of some $i \in \mathcal{CS}$ with certificate cert'_i, $\mathcal{A}$ locally checks $\Pi.\mathsf{Verify}(j, i, 1, \mathsf{mvk}_{j,i}, \mathsf{vk}_{j,i,1}, \mathsf{cert}'_i) = \top$ for every $j \in \mathcal{CS}$, and aborts if any check fails. Otherwise, it submits cert'_i to the unanimously verifiable deletion experiment, whose challenger checks the same condition for every $j \notin \mathcal{CS}$. Hence the deletion is accepted if and only if $\mathsf{Verify}'(\mathsf{vk}', i, \mathsf{cert}'_i) = \top$. If instead $i \notin \mathcal{CS}$, $\mathcal{A}$ invokes the honest-deletion branch of the unanimously verifiable deletion experiment and discards the resulting certificate.

Thus the view of $\mathcal{A}'$ is distributed exactly as in the adaptive certified deletion experiment for Π'. Therefore, any non-negligible distinguishing advantage of $\mathcal{A}'$ yields the same advantage against the adaptive unanimously verifiable deletion security of Π, a contradiction. □

3.2 Adaptively Secure Outsourced MPC with Unanimously Verifiable Deletion

In this section, we formally define adaptively secure outsourced MPC with unanimously verifiable deletion against adaptive semi-honest corruption of the clients and adaptive malicious corruption of the computing parties. We focus on the honest-majority setting, where the number n of computing parties and the corruption threshold t satisfy $n \geq 2t + 1$.

An outsourced MPC protocol Π consists of a QPT trusted preprocessing party $\mathcal{T}$, a set of n PPT clients $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$, and a set of n QPT servers (computing parties) $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$. Each client $\mathcal{C}_i$ initially holds an input $x_i \in \{0,1\}^*$, whereas the computing parties hold no private inputs of their own and are responsible for evaluating the protocol. For notational simplicity, we consider the same number n of input clients and computing parties. This equality is not essential to the model: one may instead consider any polynomial number of input clients independently of the number of computing parties.

The clients may be adaptively corrupted in the semi-honest sense. Upon corruption, the adversary learns the client's entire current classical view, including its input and preprocessing information, while the client continues to follow the prescribed protocol. At most $n - 1$ clients may be corrupted throughout the execution. The computing parties may be adaptively corrupted throughout the online execution and, once corrupted, may arbitrarily deviate from the prescribed protocol. After successful finalization, all remaining computing parties may additionally be corrupted without being subject to the online corruption bound.

The clients participate only in the input phase and retire immediately after submitting their inputs. Retirement means that they no longer participate in the protocol, although their residual classical states may still be revealed by subsequent corruptions. All online communication is conducted through a classical synchronous authenticated broadcast channel. Each client broadcasts its protocol-prescribed input message during the input phase, while all other protocol messages are broadcast by the computing parties.

Real world. The adversary $\mathcal{A}$ is a computationally unbounded interactive quantum adversary initially holding an auxiliary quantum state aux. Initialize the corrupted-client set $\mathcal{CC} \leftarrow \emptyset$, the

corrupted-server set $\mathcal{CS} \leftarrow \emptyset$, the adaptive-deletion set $\mathcal{DS} \leftarrow \emptyset$, and a flag $\mathsf{finalized} \leftarrow 0$. The real-world execution proceeds as follows.

- **Offline phase.** A trusted party $\mathcal{T}$ generates the preprocessing information prescribed by Π. For each $i \in [n]$, it gives client $\mathcal{C}_i$ a classical register cdata_i, and gives computing party $\mathcal{P}_i$ a register pdata_i together with verification keys $\{\mathsf{ver_key}_{i,j}\}_{j\in[n]}$. The concrete contents of these registers are determined by the protocol.

- **Initial corruption-and-deletion phase.** After the preprocessing information has been distributed and before the input phase begins, $\mathcal{A}$ may perform any finite sequence of the following operations:

  - *Corrupt a client.* The adversary chooses $i \notin \mathcal{CC}$, subject to $|\mathcal{CC}| < n-1$, receives the entire current classical view of $\mathcal{C}_i$, including x_i, cdata_i, local randomness, and message history, and sets $\mathcal{CC} \leftarrow \mathcal{CC} \cup \{i\}$.
  - *Corrupt a computing party.* The adversary chooses $i \notin \mathcal{CS}$ and receives the entire current view of $\mathcal{P}_i$, including its classical and quantum registers, its preprocessing information, its verification keys, and the entire history of the messages it has sent and received. It then updates $\mathcal{CS} \leftarrow \mathcal{CS} \cup \{i\}$.
  - *Delete a computing party's state.* The adversary chooses $i \notin \mathcal{DS}$. If $i \notin \mathcal{CS}$, party $\mathcal{P}_i$ follows the prescribed deletion procedure and broadcasts a classical deletion certificate $\mathsf{del_cert}_i$. If $i \in \mathcal{CS}$, the adversary broadcasts an arbitrary classical string $\mathsf{del_cert}_i$ on behalf of $\mathcal{P}_i$. Every honest computing party $\mathcal{P}_j$ verifies the certificate using $\mathsf{ver_key}_{j,i}$ and broadcasts the prescribed verification result. For every corrupted computing party, the adversary may broadcast an arbitrary message on its behalf. If any party broadcasts $\perp$, all honest computing parties abort. Otherwise, set $\mathcal{DS} \leftarrow \mathcal{DS} \cup \{i\}$.

- **Input phase.** Every client $\mathcal{C}_i$ broadcasts the input message prescribed by Π using its input x_i and preprocessing information cdata_i. At the end of this phase, all clients retire and take no further part in the protocol.

- **Online computation.** The computing parties execute the remaining rounds of Π. The protocol Π specifies a finite sequence of adaptive-query points during the online execution. Each adaptive-query point occurs between two prescribed protocol blocks, before any honest computing party has generated a message for the next block. At each such point, and before successful finalization, $\mathcal{A}$ may adaptively corrupt additional clients subject to $|\mathcal{CC}| \leq n-1$, and may perform any finite sequence of the computing-party corruption and deletion operations defined above.

- **Terminal deletion and output.** At the end of the online computation, the computing parties execute the terminal deletion-and-output phase prescribed by Π. This phase deletes all remaining secret-dependent quantum registers that the protocol requires to be deleted, verifies the corresponding deletion certificates, and reconstructs the public output. If any required certificate is rejected or any prescribed check fails, the honest computing parties output $\perp$ and abort. Otherwise, all honest computing parties obtain a common output $o \in (\{0,1\}^*)^{n'}$ and set $\mathsf{finalized} \leftarrow 1$.

- **Post-execution corruption.** If $\mathsf{finalized} = 1$, the adversary may corrupt every remaining computing party, eventually obtaining $\mathcal{CS} = [n]$. Upon corrupting $\mathcal{P}_i$ in this phase, the adversary receives only its residual post-execution view, including all remaining registers, verification keys, and message history. The adversary may also continue to corrupt clients in this phase, subject to $|\mathcal{CC}| \leq n - 1$, receiving their residual classical views.

We say that $\mathcal{A}$ is *online t-limited* if, at every point before successful finalization, it holds that

$$\left|\mathcal{CS} \setminus \mathcal{DS}\right| \leq t \quad \text{and} \quad \left|[n] \setminus \mathcal{DS}\right| - \left|\mathcal{CS} \setminus \mathcal{DS}\right| \geq t + 1.$$

In particular, when $\mathcal{DS} = \emptyset$, this condition allows at most t computing parties to be corrupted when $n \geq 2t+1$. The condition is imposed only before successful finalization; after successful finalization, all computing parties may be corrupted.

Let $\boldsymbol{x} := (x_1, \ldots, x_n)$. The output of the real-world experiment, denoted by $\mathsf{Real}^{\Pi}_{\mathcal{A}}(1^\lambda, \boldsymbol{x}, \mathsf{aux})$, consists of the following two parts:

1. the common output of the honest computing parties $o \in (\{0,1\}^*)^{n'}$, or $\perp$ if the protocol aborts; and
2. the final view of $\mathcal{A}$, including the views obtained by post-execution corruption.

Ideal world. In the ideal-world experiment, the clients interact with an ideal functionality $\mathcal{F}$ realizing $F : (\{0,1\}^*)^n \to (\{0,1\}^*)^{n'}$. The ideal-world adversary, or simulator, is denoted by $\mathcal{S}$. At the beginning of the experiment, client $\mathcal{C}_i$ holds input x_i, while $\mathcal{S}$ receives only an auxiliary quantum state aux. In particular, $\mathcal{S}$ is not initially given any client's input.

- **Adaptive client corruption.** At the same corruption points as in the real-world experiment, $\mathcal{S}$ may corrupt an uncorrupted client $\mathcal{C}_i$, subject to at most $n - 1$ corrupted clients in total. Upon corruption, $\mathcal{S}$ receives x_i and is responsible for simulating the client's current classical view.
- **Input submission.** Each uncorrupted client $\mathcal{C}_i$ submits its true input x_i directly to $\mathcal{F}$, while for each already corrupted client, $\mathcal{S}$ submits the revealed input x_i on its behalf. Each client then retires.
- **Ideal computation.** The functionality computes $o := F(\boldsymbol{x})$ and stores the result. Before the output phase, $\mathcal{S}$ may send ABORT to $\mathcal{F}$, in which case $\mathcal{F}$ outputs $\perp$ and terminates.
- **Output phase.** If no abort has occurred, $\mathcal{F}$ sends o to $\mathcal{S}$. The simulator then sends either ABORT or $\mathsf{DELIVER}$ to $\mathcal{F}$. If it sends ABORT, the output of $\mathcal{F}$ is $\perp$; if it sends $\mathsf{DELIVER}$, the output of $\mathcal{F}$ is o.

The output of the ideal-world experiment, denoted by $\mathsf{Ideal}^{F}_{\mathcal{S}}(1^\lambda, \mathbf{x}, \mathsf{aux})$, consists of the following two parts:

1. the output of $\mathcal{F}$, which is either o or $\perp$; and
2. the final simulated adversarial view output by $\mathcal{S}$.

Defining security. We now define adaptive security of outsourced MPC with unanimously verifiable deletion against adaptive semi-honest corruption of the clients and adaptive malicious corruption of the computing parties.

Definition 20 (Adaptive Security of Outsourced MPC with Unanimously Verifiable Deletion)**.** *Let n, t be positive integers such that $n \geq 2t+1$, let $F : (\{0,1\}^*)^n \longrightarrow (\{0,1\}^*)^{n'}$ be a function, and let Π be an outsourced MPC protocol computing F. We say that Π evaluates F with (t,n)-adaptively secure unanimously verifiable deletion and post-execution full-corruption security if, for every computationally unbounded quantum real-world adversary $\mathcal{A}$ that is online t-limited with respect to the computing parties and adaptively corrupts at most $n-1$ clients throughout the experiment, there exists a quantum ideal-world simulator $\mathcal{S}$ such that*

$$\mathsf{TD}\Big(\mathsf{Real}^{\Pi}_{\mathcal{A}}(1^\lambda, \boldsymbol{x}, \mathsf{aux}), \mathsf{Ideal}^{F}_{\mathcal{S}}(1^\lambda, \boldsymbol{x}, \mathsf{aux})\Big) \leq \mathsf{negl}(\lambda)$$

for every input vector $\boldsymbol{x} \in (\{0,1\}^*)^n$ and every auxiliary state $\mathsf{aux} \in \mathcal{H}^*$.

Remark 21 (Round Adaptivity)**.** *Our notion of adaptive security should be interpreted as round-adaptive security. During the online execution, the adversary may issue client-corruption, computing-party-corruption, and deletion queries only at the designated adaptive-query points specified by the security experiment: initially after the offline phase and before the input phase, and subsequently between prescribed communication rounds. In particular, it cannot issue such a query after some honest computing parties have generated or broadcast their messages for a round but before the remaining honest computing parties have done so.*

This restriction is necessary because verifying a deletion certificate is itself an interactive subprotocol that requires communication among the computing parties. Allowing a deletion query in the middle of another communication round would require the unanimous-verification subprotocol to interrupt the ongoing round, making the execution order and the parties' local states unclear. Accordingly, all adaptive queries during the online execution are allowed only at these designated points.

Remark 22 (Security Setting)**.** *We clarify the security setting of our outsourced MPC protocol using the standard terminology in the MPC literature.*

- ***Semi-honest classical clients.*** *The clients are classical PPT machines. They receive only classical preprocessing information, provide their private inputs through classical messages, and never receive or manipulate quantum states. They may be adaptively corrupted in the semi-honest sense: corruption reveals the client's input and complete current classical view, but the client continues to follow the prescribed protocol. They do not participate in circuit evaluation, deletion verification, or output reconstruction, and retire immediately after the input phase. Their residual classical views may nevertheless be revealed by later corruptions. At most $n-1$ clients may be corrupted throughout the experiment.*
- ***Classical synchronous broadcast channel.*** *All online communication by both the input clients and the computing parties is conducted exclusively through a classical synchronous authenticated broadcast channel. Each client uses this channel to submit its prescribed input message, while the computing parties use the same channel for all other protocol messages. "Synchronous" means that every prescribed message must be received within the corresponding communication round. If an honest computing party does not receive a required message by the end of the round, it may immediately abort the protocol.*

- ***Malicious quantum computing parties.*** *The computing parties are* QPT *machines and may hold and manipulate quantum states throughout the protocol. Corrupted computing parties may arbitrarily deviate from the prescribed protocol, submit invalid deletion certificates, send incorrect protocol messages, or cause the protocol to abort.*

- ***Rushing adversary.*** *In each computing-party communication round, the adversary may first observe all messages broadcast by honest computing parties before determining the messages broadcast on behalf of corrupted computing parties in the same round.*

- ***Trusted preprocessing.*** *The protocol is formulated in the trusted-preprocessing model. Before the online execution, a trusted party distributes the required preprocessing information to both the input clients and the computing parties.*

- ***Online honest-majority condition.*** *During the online execution and before successful finalization, security holds provided that*

$$|\mathcal{CS} \setminus \mathcal{DS}| \leq t \qquad \text{and} \qquad |[n] \setminus \mathcal{DS}| - |\mathcal{CS} \setminus \mathcal{DS}| \geq t + 1.$$

  When no computing party has deleted its state, these conditions reduce to the usual bound $|\mathcal{CS}| \leq t$, with the second condition following from $n \geq 2t + 1$.

- ***Post-execution full corruption.*** *Once the terminal deletion-and-output phase has successfully completed, the online computing-party corruption bound is removed and the adversary may corrupt all computing parties. Client corruptions may also continue, subject to at most $n - 1$ corrupted clients in total. The inputs of the remaining uncorrupted clients must remain private beyond what is revealed by the prescribed output and the inputs of the corrupted clients.*

- ***Public reconstruction.*** *At the end of a successful execution, a common output is publicly reconstructed and revealed to all entities.*

- ***Security with abort.*** *A malicious computing-party adversary may cause the execution to abort, in which case the honest entities terminate without receiving an output.*

4 Construction of Linearly Homomorphic Secret-Sharing Batch with Unanimously Verifiable Deletion

In this section, we construct an $(n, t, M, \mathbb{Z}_q)$ secret-sharing batch scheme $\Pi^{\mathsf{SSB\text{-}UVD}}_{n,t,M,\mathbb{Z}_q}$ with unanimously verifiable deletion. We begin in Section 4.1 by presenting the linear-homomorphic structure of BB84-type states underlying our construction. We then specify the parameters in Section 4.2 and describe the scheme in Section 4.3. Finally, we establish its correctness and security in Sections 4.4 and 4.5, respectively.

4.1 A Linearly Homomorphic Structure of BB84-Type States with a Common Basis String

In this section, we present a linear-homomorphic structure of BB84-type states. Informally, if two such states share the same basis string $\boldsymbol{\theta}$, then the local unitary $\mathbf{U}^m_{\mathcal{C}}$ preserves the family of BB84-type states and implements an affine linear update on their labels.

Let q be a prime. For any $c \in \mathbb{Z}_q$ and $m \in \mathbb{N}^*$, let $\mathbf{U}_c^m$ be the unitary operator on $\mathbb{C}^{(\mathbb{Z}_q^m)^{\otimes 2}}$ defined by its action on computational basis states:

$$\mathbf{U}_c^m : |\mathbf{a}\rangle|\mathbf{b}\rangle \longmapsto |\mathbf{a}\rangle|c\mathbf{a}+\mathbf{b}\rangle, \quad \forall \mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^m \tag{5}$$

When $c = 1$ or $m = 1$, we may omit the corresponding subscript or superscript, writing $\mathbf{U}^m$, $\mathbf{U}_c$, or simply $\mathbf{U}$. For $\mathbf{x} \in \mathbb{Z}_q^m$ and $\boldsymbol{\theta} \in \{0,1\}^m$, define

$$|\mathbf{x}^{\boldsymbol{\theta}}\rangle := \bigotimes_{j=1}^{m} \mathsf{QFT}^{\theta[j]}|x[j]\rangle.$$

where $\mathsf{QFT}^{\theta[j]}$ denotes QFT if $\theta[j] = 1$ or the identity operation $\mathbf{I}$ if $\theta[j] = 0$.

Theorem 23 (Linear Update of BB84-Type States with a Common Basis String)**.** *Let $\boldsymbol{\theta} \in \{0,1\}^m$ and let $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{Z}_q^m$. For any $c \in \mathbb{Z}_q$, define $\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2 \in \mathbb{Z}_q^m$ coordinate-wise by*

$$\widetilde{x}_1[j] := \begin{cases} x_1[j], & \theta[j] = 0, \\ x_1[j] - c\,x_2[j], & \theta[j] = 1, \end{cases} \qquad \widetilde{x}_2[j] := \begin{cases} c\,x_1[j] + x_2[j], & \theta[j] = 0, \\ x_2[j], & \theta[j] = 1. \end{cases}$$

Then

$$\mathbf{U}_c^m\big(|\mathbf{x}_1^{\boldsymbol{\theta}}\rangle \otimes |\mathbf{x}_2^{\boldsymbol{\theta}}\rangle\big) = |\widetilde{\mathbf{x}}_1^{\boldsymbol{\theta}}\rangle \otimes |\widetilde{\mathbf{x}}_2^{\boldsymbol{\theta}}\rangle.$$

Proof. Since both the state and the unitary decompose coordinate-wise, it suffices to analyze one coordinate j.

- If $\theta[j] = 0$, then the j-th tensor factor is $|x_1[j]\rangle \otimes |x_2[j]\rangle$. By the definition of $\mathbf{U}_c$, we have

$$\mathbf{U}_c\big(|x_1[j]\rangle \otimes |x_2[j]\rangle\big) = |x_1[j]\rangle \otimes |c\,x_1[j] + x_2[j]\rangle.$$

- If $\theta[j] = 1$, then the j-th tensor factor is $\mathsf{QFT}|x_1[j]\rangle \otimes \mathsf{QFT}|x_2[j]\rangle$. By Lemma 6,

$$\mathbf{U}_c\big(\mathsf{QFT}|x_1[j]\rangle \otimes \mathsf{QFT}|x_2[j]\rangle\big) = \mathsf{QFT}|x_1[j] - c\,x_2[j]\rangle \otimes \mathsf{QFT}|x_2[j]\rangle,$$

□

Applications beyond secret sharing. The linear structure above is not specific to secret sharing. In Appendix A, we illustrate its broader applicability by constructing a linearly homomorphic secret-key encryption scheme with certified deletion. We also discuss how the same technique can be applied more generally to the BB84-based certified-deletion compilers of [BK23] to preserve linear homomorphism for certain underlying primitives, such as public-key encryption, by generating multiple ciphertexts with the same freshly sampled basis string $\boldsymbol{\theta}$ so that they can be evaluated homomorphically while retaining certified deletion.

4.2 Parameters

Our construction uses the following parameters:

- λ denotes the security parameter.
- $q = 2^{\Theta(\lambda)}$ denotes the prime modulus.
- $n \in \text{poly}(\lambda)$ denotes the number of parties.
- $t \in [1, n-1]$ denotes the maximum number of corrupted shares that may remain undeleted at any time.
- $m := 4n(n-t)(n+2)^2\lambda$ denotes the length of every party's share.
- $\ell := 4(n-t)^2\lambda$ denotes the number of checking points contributed by each party to every other party's share. (Hence, each share contains a total of $n\ell$ checking points.)
- $m' := m - n\ell$ denotes the number of evaluation points in each share.
- $m'_L := m\sqrt{\frac{\lambda}{\ell}} = 2n(n+2)^2\lambda$ intuitively represents an upper bound on the amount of information which is not destroyed when an adversary generates a valid deletion certificate for a share. (We will show the specific use of this parameter later.)
- $m'_R := m' - m'_L$.
- $d := nm'_L + tm'_R$ denotes the degree of the polynomial used in the construction.
- $M \in \text{poly}(\lambda)$ denotes the number of secrets in each batch.

Note that this parameter setting ensures exponential security. If only superpolynomial security is required, let $p : \mathbb{N} \to \mathbb{N}$ be any integer-valued function satisfying $p(\lambda) = \omega(\log \lambda)$, and replace λ by $p(\lambda)$ in the definitions of m, ℓ, and m'_L above.

We remark that the notations m', m'_L, and m'_R do not explicitly appear in our construction, but are important in our proof.

4.3 Construction

In the following we describe the scheme $\Pi^{\mathsf{SSB-UVD}}_{n,t,M,\mathbb{Z}_q}$.

- $\mathsf{Setup}(1^\lambda)$: For every $i_2 \in [n]$, sample $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1\in[n]}$ as n pairwise disjoint subsets of $[m]$, with $|\mathsf{mvk}_{i_1,i_2}| = \ell$ for each $i_1 \in [n]$. Output $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]}$. [2]
- $\mathsf{Split}(\iota \in [M], s_\iota \in \mathbb{Z}_q, \{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]})$: Sample $\sigma_\iota \leftarrow \mathbb{Z}_q$ uniformly at random, and sample a polynomial f_ι of degree at most d such that $f_\iota(0) = s_\iota - \sigma_\iota$. For every $i_1 \in [n]$, $j \in [m]$,

[2]Note that the master verification keys owned by $\mathcal{P}_{i_1}$ are actually $\{\mathsf{mvk}_{i_1,i_2}\}_{i_2\in[n]}$, although the setup procedure samples $\{\mathsf{mvk}_{i_1,i_2}\}_{i_1\in[n]}$ jointly for each fixed i_2.

- If $j \in \mathsf{mvk}_{i_2,i_1}$ for some $i_2 \in [n]$ (j is a checking point), then sample $r_{i_1,j,\iota} \leftarrow \mathbb{Z}_q$, and set $|\psi_{i_1,j,\iota}\rangle := \mathsf{QFT}|r_{i_1,j,\iota}\rangle$.
- Otherwise (j is an evaluation point), set $|\psi_{i_1,j,\iota}\rangle := |f_\iota((i_1-1)m+j)\rangle$.

Let $|\boldsymbol{\psi}_{i_1,\iota}\rangle := \bigotimes_{j=1}^m |\psi_{i_1,j,\iota}\rangle$. Let $\mathsf{sh}_{i_1,\iota} := (|\boldsymbol{\psi}_{i_1,\iota}\rangle, \sigma_\iota)$. After generating $\{\mathsf{sh}_{i,\iota}\}_{i\in[n]}$, define $\mathsf{vk}_{i_1,i_2,\iota} := \{(j, r_{i_2,j,\iota})\}_{j\in\mathsf{mvk}_{i_1,i_2}}$ for every $i_1, i_2 \in [n]$. Output $(\{\mathsf{vk}_{i_1,i_2,\iota}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}_{i,\iota}\}_{i\in[n]})$.

- $\mathsf{AddSh}(c \in \mathbb{Z}_q, i \in [n], \iota \in [M], \mathsf{sh}_{i,\iota})$: Parse $\mathsf{sh}_{i,\iota} = (|\boldsymbol{\psi}_{i,\iota}\rangle, \sigma_\iota)$. Output $\widetilde{\mathsf{sh}}_{i,\iota} = (|\boldsymbol{\psi}_{i,\iota}\rangle, \sigma_\iota + c)$.
- $\mathsf{ScaleAddSh}(c \in \mathbb{Z}_q, i \in [n], \iota_1 \in [M], \iota_2 \in [M], \mathsf{sh}_{i,\iota_1}, \mathsf{sh}_{i,\iota_2})$: Parse $\mathsf{sh}_{i,\iota_1} = (|\boldsymbol{\psi}_{i,\iota_1}\rangle, \sigma_{\iota_1})$ and $\mathsf{sh}_{i,\iota_2} = (|\boldsymbol{\psi}_{i,\iota_2}\rangle, \sigma_{\iota_2})$. Let $\mathbf{U}_c^m$ be defined as in Eqn (5). Let $|\tilde{\boldsymbol{\psi}}_{i,\iota_1}\rangle|\tilde{\boldsymbol{\psi}}_{i,\iota_2}\rangle := \mathbf{U}_c^m|\boldsymbol{\psi}_{i,\iota_1}\rangle|\boldsymbol{\psi}_{i,\iota_2}\rangle$. Output $\widetilde{\mathsf{sh}}_{i,\iota_1} := (|\tilde{\boldsymbol{\psi}}_{i,\iota_1}\rangle, \sigma_{\iota_1}), \widetilde{\mathsf{sh}}_{i,\iota_2} := (|\tilde{\boldsymbol{\psi}}_{i,\iota_2}\rangle, c\cdot\sigma_{\iota_1} + \sigma_{\iota_2})$.
- $\mathsf{ScaleAddVk}(c \in \mathbb{Z}_q, i_1 \in [n], i_2 \in [n], \iota_1 \in [M], \iota_2 \in [M], \mathsf{mvk}_{i_1,i_2}, \mathsf{vk}_{i_1,i_2,\iota_1}, \mathsf{vk}_{i_1,i_2,\iota_2})$: Parse $\mathsf{vk}_{i_1,i_2,\iota_1} := \{(j, r_{i_2,j,\iota_1})\}_{j\in\mathsf{mvk}_{i_1,i_2}}, \mathsf{vk}_{i_1,i_2,\iota_2} := \{(j, r_{i_2,j,\iota_2})\}_{j\in\mathsf{mvk}_{i_1,i_2}}$. Output $\widetilde{\mathsf{vk}}_{i_1,i_2,\iota_1} := \{(j, r_{i_2,j,\iota_1} - c\cdot r_{i_2,j,\iota_2})\}_{j\in\mathsf{mvk}_{i_1,i_2}}, \widetilde{\mathsf{vk}}_{i_1,i_2,\iota_2} := \mathsf{vk}_{i_1,i_2,\iota_2}$.
- $\mathsf{Reconstruct}(\mathcal{IP}, \mathcal{IS}, \{\mathsf{sh}_{i,\iota}\}_{i\in\mathcal{IP},\iota\in\mathcal{IS}}) \to (\{s_\iota\}_{\iota\in\mathcal{IS}}/\perp)$: If $|\mathcal{IP}| \le t$, output $\perp$. Otherwise, for every $i \in \mathcal{IP}, \iota \in \mathcal{IS}$, parse $\mathsf{sh}_{i,\iota} = (|\boldsymbol{\psi}_{i,\iota}\rangle, \sigma_\iota)$, and measure $|\boldsymbol{\psi}_{i,\iota}\rangle$ on the computational basis to get $\mathbf{x}_{i,\iota} \in \mathbb{Z}_q^m$. Apply $\mathsf{Correct}_{\mathbb{Z}_q,d}$ (Lemma 8) to $\{((i-1)m+j, x_{i,\iota}[j])\}_{i\in\mathcal{IP},j\in[m]}$. If it outputs $\perp$, output $\perp$. Otherwise, let f_ι be the returned polynomial and compute $s_\iota := f_\iota(0) + \sigma_\iota$.
- $\mathsf{Delete}(i, \iota, \mathsf{sh}_{i,\iota})$: Parse $\mathsf{sh}_{i,\iota} = (|\boldsymbol{\psi}_{i,\iota}\rangle, \sigma_\iota)$, and measure $|\boldsymbol{\psi}_{i,\iota}\rangle$ on the QFT basis to get $\mathsf{cert}_{i,\iota} \in \mathbb{Z}_q^m$. Output $\mathsf{cert}_{i,\iota}$.
- $\mathsf{Verify}(i_1, i_2, \iota, \mathsf{mvk}_{i_1,i_2}, \mathsf{vk}_{i_1,i_2,\iota}, \mathsf{cert}_{i_2,\iota})$: Parse $\mathsf{vk}_{i_1,i_2,\iota} = \{(j, r_{i_2,j,\iota})\}_{j\in\mathsf{mvk}_{i_1,i_2}}$. Check whether $r_{i_2,j,\iota} = \mathsf{cert}_{i_2,\iota}[j]$ for every $j \in \mathsf{mvk}_{i_1,i_2}$. If so, output $\top$; otherwise, output $\perp$.

4.4 Correctness

In this section, we prove the reconstruction correctness, deletion correctness, and linear homomorphism of the scheme in section 4.3. To this end, we first introduce some notations that will be used throughout the proofs both in Section 4.4 and in Section 4.5.

For each $i_1 \in [n]$, let

$$\mathcal{J}_{i_1}^{\checkmark} := [m] \backslash \Big(\bigcup_{i_2\in[n]} \mathsf{mvk}_{i_2,i_1} \Big) \tag{6}$$

be the set of evaluation points, and

$$\mathcal{J}_{i_1}^{\times} := \bigcup_{i_2\in[n]} \mathsf{mvk}_{i_2,i_1} \tag{7}$$

be the set of checking points.

The following theorem establishes the reconstruction correctness of $\Pi_{n,t,M,\mathbb{Z}_q}^{\mathsf{SSB-UVD}}$.

Theorem 24. $\Pi_{n,t,M,\mathbb{Z}_q}^{\mathsf{SSB-UVD}}$ *satisfies reconstruction correctness.*

Proof. Fix any $\mathcal{IP}$ with $|\mathcal{IP}| \geq t+1$. For each $i_1 \in \mathcal{IP}$, the equality

$$x_{i_1,\iota}[j] = f_\iota((i_1-1)m+j)$$

fails only when $j \in \mathcal{J}_{i_1}^{\times}$. Therefore, among the points

$$\{((i_1-1)m+j,\, x_{i_1,\iota}[j])\}_{i_1 \in \mathcal{IP},\, j \in [m]},$$

the number of erroneous evaluations is at most

$$\sum_{i_1 \in \mathcal{IP}} |\mathcal{J}_{i_1}^{\times}|.$$

By Lemma 8, the polynomial f_ι can be uniquely recovered provided that

$$2 \cdot \sum_{i_1 \in \mathcal{IP}} |\mathcal{J}_{i_1}^{\times}| < |\mathcal{IP}| \cdot m - d.$$

Since $|\mathcal{J}_{i_1}^{\times}| = n\ell$ for every i_1, this condition is equivalent to

$$d < |\mathcal{IP}| \cdot (m - 2n\ell).$$

Since $|\mathcal{IP}| \geq t+1$, it suffices to show

$$d < (t+1)(m-2n\ell).$$

By the definitions of d and m'_R,

$$d = (n-t)m'_L + t(m-n\ell).$$

Since

$$m'_L = m\sqrt{\frac{\lambda}{\ell}} = \frac{m}{2(n-t)},$$

we have

$$(n-t)m'_L = \frac{m}{2}.$$

Therefore, it suffices to show

$$(t+2)n\ell < \frac{m}{2}.$$

Substituting $\ell = 4(n-t)^2\lambda$ and $m = 4n(n-t)(n+2)^2\lambda$, this is equivalent to

$$2(t+2)(n-t) < (n+2)^2.$$

This holds since

$$(t+2)(n-t) \leq \frac{(n+2)^2}{4}.$$

Hence $d < (t+1)(m-2n\ell)$, so f_ι can be uniquely recovered, and therefore the secret is reconstructed correctly. □

The following theorem establishes the deletion correctness of $\Pi^{\mathsf{SSB-UVD}}_{n,t,M,\mathbb{Z}_q}$.

Theorem 25. $\Pi^{\mathsf{SSB-UVD}}_{n,t,M,\mathbb{Z}_q}$ *satisfies deletion correctness.*

Proof. This follows immediately from the construction. For every $i_1, i_2 \in [n]$, $\iota \in [M]$, and $j \in \mathsf{mvk}_{i_1,i_2}$, the corresponding state is prepared as $|\psi_{i_2,j,\iota}\rangle = \mathsf{QFT}|r_{i_2,j,\iota}\rangle$. Therefore, when party $\mathcal{P}_{i_2}$ measures it in the QFT basis, the outcome is exactly $r_{i_2,j,\iota}$. Hence every honest verification check accepts. □

Theorem 26. $\Pi^{\mathsf{SSB-UVD}}_{n,t,M,\mathbb{Z}_q}$ *satisfies linear homomorphism.*

Proof. We prove the theorem by showing that the following invariant is preserved under both AddSh and $\mathsf{ScaleAddSh/ScaleAddVk}$. Fix any batch state St and secret vector $\mathbf{s}$ such that $\mathsf{Reach}(\mathsf{St}, \mathbf{s})$. By the definition of Reach and the construction of Split, there exist polynomials $\{f_\iota\}_{\iota\in[M]}$ of degree at most d, classical values $\{\sigma_\iota\}_{\iota\in[M]}$, and checking values $\{r_{i,j,\iota}\}_{i\in[n],j\in\mathcal{J}_i^\times,\iota\in[M]}$ satisfying the following conditions. For every $i \in [n]$, define the basis string $\boldsymbol{\theta}_i \in \{0,1\}^m$ by

$$\theta_i[j] := \begin{cases} 0, & j \in \mathcal{J}_i^\checkmark, \\ 1, & j \in \mathcal{J}_i^\times. \end{cases}$$

For every $i \in [n]$ and $\iota \in [M]$, define $\boldsymbol{x}_{i,\iota} \in \mathbb{Z}_q^m$ coordinate-wise by

$$x_{i,\iota}[j] := \begin{cases} f_\iota((i-1)m + j), & j \in \mathcal{J}_i^\checkmark, \\ r_{i,j,\iota}, & j \in \mathcal{J}_i^\times. \end{cases}$$

Then

$$\mathsf{sh}_{i,\iota} = \left(\left|\boldsymbol{x}_{i,\iota}^{\boldsymbol{\theta}_i}\right\rangle, \sigma_\iota\right), \qquad s_\iota = f_\iota(0) + \sigma_\iota,$$

and, for every $i_1, i_2 \in [n]$,

$$\mathsf{vk}_{i_1,i_2,\iota} = \{(j, r_{i_2,j,\iota})\}_{j\in\mathsf{mvk}_{i_1,i_2}}.$$

The invariant holds for every freshly generated batch state by the definition of Split. We next show that it is preserved by each additive operation.

First, consider an operation $\mathsf{AddSh}(c, i, \iota, \mathsf{sh}_{i,\iota})$ applied to every $i \in [n]$. This operation leaves $|\boldsymbol{x}_{i,\iota}^{\boldsymbol{\theta}_i}\rangle$, f_ι, and all checking values and verification keys unchanged, and replaces σ_ι with $\widetilde{\sigma}_\iota := \sigma_\iota + c$. Therefore, the secret associated with coordinate ι becomes

$$\widetilde{s}_\iota = f_\iota(0) + \widetilde{\sigma}_\iota = f_\iota(0) + \sigma_\iota + c = s_\iota + c,$$

while all other secrets remain unchanged. Hence the resulting state is honestly reachable and is associated with the required updated secret vector.

Next, consider an operation $\mathsf{ScaleAddSh/ScaleAddVk}$ with constant $c \in \mathbb{Z}_q$ and distinct indices $\iota_1, \iota_2 \in [M]$. Fix any party $i \in [n]$. Since the master verification keys are shared across the batch, the two states $|\boldsymbol{x}_{i,\iota_1}^{\boldsymbol{\theta}_i}\rangle$ and $|\boldsymbol{x}_{i,\iota_2}^{\boldsymbol{\theta}_i}\rangle$ have the same basis string $\boldsymbol{\theta}_i$. By Theorem 23,

$$\mathbf{U}_c^m\left(\left|\boldsymbol{x}_{i,\iota_1}^{\boldsymbol{\theta}_i}\right\rangle\left|\boldsymbol{x}_{i,\iota_2}^{\boldsymbol{\theta}_i}\right\rangle\right) = \left|\widetilde{\boldsymbol{x}}_{i,\iota_1}^{\boldsymbol{\theta}_i}\right\rangle\left|\widetilde{\boldsymbol{x}}_{i,\iota_2}^{\boldsymbol{\theta}_i}\right\rangle,$$

where, for every $j \in [m]$,

$$\widetilde{x}_{i,\iota_1}[j] = \begin{cases} x_{i,\iota_1}[j], & j \in \mathcal{J}_i^\checkmark, \\ x_{i,\iota_1}[j] - c x_{i,\iota_2}[j], & j \in \mathcal{J}_i^\times, \end{cases}$$

and

$$\widetilde{x}_{i,\iota_2}[j] = \begin{cases} cx_{i,\iota_1}[j] + x_{i,\iota_2}[j], & j \in \mathcal{J}_i^{\checkmark}, \\ x_{i,\iota_2}[j], & j \in \mathcal{J}_i^{\times}. \end{cases}$$

Define

$$\widetilde{f}_{\iota_1} := f_{\iota_1}, \qquad \widetilde{f}_{\iota_2} := cf_{\iota_1} + f_{\iota_2}.$$

For every $i \in [n]$ and $j \in \mathcal{J}_i^{\times}$, define

$$\widetilde{r}_{i,j,\iota_1} := r_{i,j,\iota_1} - cr_{i,j,\iota_2}, \qquad \widetilde{r}_{i,j,\iota_2} := r_{i,j,\iota_2}.$$

It follows from the above coordinate-wise description that, at every evaluation point,

$$\widetilde{x}_{i,\iota_1}[j] = \widetilde{f}_{\iota_1}((i-1)m + j), \qquad \widetilde{x}_{i,\iota_2}[j] = \widetilde{f}_{\iota_2}((i-1)m + j),$$

and, at every checking point,

$$\widetilde{x}_{i,\iota_1}[j] = \widetilde{r}_{i,j,\iota_1}, \qquad \widetilde{x}_{i,\iota_2}[j] = \widetilde{r}_{i,j,\iota_2}.$$

Moreover, $\mathsf{ScaleAddVk}$ updates the corresponding verification keys to

$$\widetilde{\mathsf{vk}}_{i_1,i_2,\iota_1} = \{(j, \widetilde{r}_{i_2,j,\iota_1})\}_{j \in \mathsf{mvk}_{i_1,i_2}}, \qquad \widetilde{\mathsf{vk}}_{i_1,i_2,\iota_2} = \{(j, \widetilde{r}_{i_2,j,\iota_2})\}_{j \in \mathsf{mvk}_{i_1,i_2}},$$

so the verification keys remain consistent with the checking-point labels of the transformed shares. Finally, $\mathsf{ScaleAddSh}$ sets

$$\widetilde{\sigma}_{\iota_1} := \sigma_{\iota_1}, \qquad \widetilde{\sigma}_{\iota_2} := c\sigma_{\iota_1} + \sigma_{\iota_2}.$$

Consequently, the secrets associated with the transformed shares are

$$\widetilde{s}_{\iota_1} = \widetilde{f}_{\iota_1}(0) + \widetilde{\sigma}_{\iota_1} = f_{\iota_1}(0) + \sigma_{\iota_1} = s_{\iota_1}$$

and

$$\widetilde{s}_{\iota_2} = \widetilde{f}_{\iota_2}(0) + \widetilde{\sigma}_{\iota_2} = c(f_{\iota_1}(0) + \sigma_{\iota_1}) + f_{\iota_2}(0) + \sigma_{\iota_2} = cs_{\iota_1} + s_{\iota_2}.$$

All other shares, verification keys, and secrets remain unchanged. Since $\widetilde{f}_{\iota_1}$ and $\widetilde{f}_{\iota_2}$ have degree at most d, the transformed values above can arise from valid choices of the random coins of Split. Therefore, $\mathsf{Reach}(\widetilde{\mathsf{St}}, \widetilde{\mathbf{s}})$ holds, where

$$\widetilde{\mathsf{St}} = \left(\{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2 \in [n]}, \{\widetilde{\mathsf{vk}}_{i_1,i_2,\iota}\}_{i_1,i_2 \in [n], \iota \in [M]}, \{\widetilde{\mathsf{sh}}_{i,\iota}\}_{i \in [n], \iota \in [M]} \right).$$

□

4.5 Security

In this section we prove the following theorem.

Theorem 27. $\Pi_{n,t,M,\mathbb{Z}_q}^{\mathsf{SSB-UVD}}$ *has adaptive unanimously verifiable deletion security.*

Proof of Theorem 27. We prove Theorem 27 via a hybrid argument. Without loss of generality, every execution that does not abort ends with $\mathcal{CS} = \mathcal{DS} = [n]$. Indeed, when $\mathcal{A}$ would otherwise terminate, it may retain its current view, honestly delete all remaining honest undeleted shares, corrupt all remaining parties, and then request deletion of the remaining undeleted shares. These requests respect $|\mathcal{CS} \setminus \mathcal{DS}| \le t$; the final deletions are accepted because no honest verifier remains. Discarding the appended information recovers the original view.

For any $b \in \{0,1\}$ and any two batches of secrets $\{s_\iota^{(0)}\}_{\iota\in[M]}$ and $\{s_\iota^{(1)}\}_{\iota\in[M]}$, we define a sequence of hybrids and establish their indistinguishability through a series of lemmas. We briefly describe the hybrids below, whose formal definitions will be given afterward.

- $\mathsf{Hyb}_0(b)$: This is the real-world game with secret batch $\{s_\iota^{(b)}\}_{\iota\in[M]}$.
- $\mathsf{Hyb}_1(b)$: When an honest party deletes its share, the challenger generates the certificate using the verification values and fresh randomness, without measuring the share.
- $\mathsf{Hyb}_2(b)$: The challenger generates the data values lazily. It samples all L values and the R values of the first t quantum shares delivered to the adversary uniformly at random. All later R values are interpolated from these fixed values and the value at 0.
- $\mathsf{Hyb}_3(b)$: The challenger purifies this generation procedure and retains the private registers used for interpolation.
- $\mathsf{Hyb}_{4,\tau'}(b)$ for $\tau' \in [0,n]$: Starting from $\mathsf{Hyb}_{4,0}(b) := \mathsf{Hyb}_3(b)$, the challenger adds a global certificate-dependent projection at each of the first τ' corrupt-then-delete events, after ordinary verification accepts and only if $|\mathcal{CS}| < n$.
- $\mathsf{Sim}_{4,\tau'}$ for $\tau' \in [1,n]$: Each delivered data register is half of a fresh EPR pair, and each additional check uses the corresponding local inverse-Fourier projector. These simulators are independent of the secret batch.
- $\overline{\mathsf{Hyb}}_{4,\tau'}(b)$ and $\overline{\mathsf{Sim}}_{4,\tau'}$: These experiments stop before the first corruption with $\eta(\tau) > \tau'$, or otherwise immediately after the n-th corruption. If the experiment terminates earlier, they return its view at termination.

To prove security, we combine the following lemmas and sum the errors over the n adjacent transitions for each $b \in \{0,1\}$:

$$
\begin{aligned}
\mathsf{TD}(\mathsf{Hyb}_0(0), \mathsf{Hyb}_0(1)) &= \mathsf{TD}(\mathsf{Hyb}_1(0), \mathsf{Hyb}_1(1)) && \text{(Lemma 28)}\\
&= \mathsf{TD}(\mathsf{Hyb}_2(0), \mathsf{Hyb}_2(1)) && \text{(Lemma 29)}\\
&= \mathsf{TD}(\mathsf{Hyb}_3(0), \mathsf{Hyb}_3(1)) && \text{(Lemma 30)}\\
&= \mathsf{TD}(\mathsf{Hyb}_{4,0}(0), \mathsf{Hyb}_{4,0}(1))\\
&\le \mathsf{TD}(\mathsf{Hyb}_{4,n}(0), \mathsf{Hyb}_{4,n}(1)) + 2^{-\Omega(\lambda)} && \text{(Lemma 33)}\\
&\le \mathsf{TD}(\overline{\mathsf{Hyb}}_{4,n}(0), \overline{\mathsf{Hyb}}_{4,n}(1)) + 2^{-\Omega(\lambda)} && \text{(Eqn (11))}\\
&= \mathsf{TD}(\overline{\mathsf{Sim}}_{4,n}, \overline{\mathsf{Sim}}_{4,n}) + 2^{-\Omega(\lambda)} && \text{(Lemma 32)}\\
&= 2^{-\Omega(\lambda)}.
\end{aligned}
\tag{8}
$$

□

We begin by defining $\mathsf{Hyb}_0(b)$ as follows.

- $\mathsf{Hyb}_0(b)$: the real world game $\mathsf{Exp}^{\mathsf{ss\text{-}uvd}}_{\Pi^{\mathsf{SSB-UVD}}_{n,t,M,\mathbb{Z}_q},\lambda,|\phi\rangle,\mathcal{A}}\big(\{s_\iota^{(b)}\}_{\iota\in[M]}\big)$.

To define $\mathsf{Hyb}_1(b)$, we introduce the following notation. For $\tau \in [n]$, let $i_\tau^{(C)}$ denote the index of the τ-th corrupted party, if this corruption occurs. This order includes corruptions after honest deletion. For $\tau' \in [n]$, let $i_{\tau'}^{(D)}$ denote the index of the party in the τ'-th corrupt-then-delete event, if this event occurs. Honest deletions do not advance the latter index, so there may be fewer than n corrupt-then-delete events even when all parties eventually delete their shares. Throughout, party indices refer only to events that occur; an additional check scheduled at an event that does not occur is not performed. For each $i \in [n]$, define

$$\varphi(i) := \begin{cases} 1, & \text{if corruption of } \mathcal{P}_i \text{ delivers its quantum share,} \\ 0, & \text{otherwise.} \end{cases}$$

For $\tau = 0$ and each corruption index $\tau \in [n]$ that occurs, define

$$\eta(\tau) := \max\left\{0, \sum_{\upsilon=1}^{\tau} \varphi(i_\upsilon^{(C)}) - t\right\}.$$

Thus $\eta(\tau)$ is the number of quantum shares delivered beyond the first t during the first τ corruptions. Let $\hat{\tau} \in [n]$ be the corruption index at which the adversary receives its t-th quantum share, if such a delivery occurs, and set $\hat{\tau} := n$ otherwise.

The following observations are immediate.

- At every corruption with $\tau \in [1, \hat{\tau}]$, we have $\eta(\tau) = 0$.
- At every legal corruption with $\tau \in (\hat{\tau}, n]$ and $\varphi(i_\tau^{(C)}) = 1$, the $\eta(\tau)$-th corrupt-then-delete event has already occurred. Otherwise, after this corruption the adversary would hold more than t undeleted shares.
- At every corruption with $\tau \in (\hat{\tau}, n]$ and $\varphi(i_\tau^{(C)}) = 0$, we have $\eta(\tau) = \eta(\tau - 1)$.

Now, we define $\mathsf{Hyb}_1(b)$ as follows:

- $\mathsf{Hyb}_1(b)$: it is the same as $\mathsf{Hyb}_0(b)$ except that whenever an honest party is instructed to delete its share, the challenger $\mathcal{C}$ does not generate the corresponding deletion certificate by honestly measuring the share. Instead, $\mathcal{C}$ generates the certificate using the verification values and fresh randomness.

  More concretely, whenever an honest party $\mathcal{P}_{i_1}$ deletes its share, for each $\iota \in [M]$, the challenger $\mathcal{C}$ generates $\mathsf{cert}_{i_1,\iota} \in \mathbb{Z}_q^m$ as follows:

  1. For each $j \in \mathcal{J}_{i_1}^{\times}$, let $i_2 \in [n]$ be the unique index such that $j \in \mathsf{mvk}_{i_2,i_1}$. We let $\mathsf{cert}_{i_1,\iota}[j] = \mathsf{vk}_{i_2,i_1,\iota}[j]$.

2. For each $j \in \mathcal{J}_{i_1}^{\checkmark}$, sample $\mathsf{cert}_{i_1,\iota}[j] \leftarrow \mathbb{Z}_q$ uniformly at random.

The certificate entries on data coordinates are independent of the polynomial values. No quantum share of this party is delivered upon a later corruption.

Lemma 28. $\mathsf{TD}(\mathsf{Hyb}_0(b), \mathsf{Hyb}_1(b)) = 0$.

Proof. The claim follows immediately from the construction. Indeed, $\mathcal{C}$ simulates exactly the distribution obtained by measuring $|\boldsymbol{\psi}_{i_1,\iota}\rangle$ in the QFT basis: on coordinates in $\mathcal{J}_{i_1}^{\times}$, the outcome is fixed by the corresponding verification values, while on coordinates in $\mathcal{J}_{i_1}^{\checkmark}$, the outcomes are independent and uniform over $\mathbb{Z}_q$, independently of the polynomial values and the adversary's state. Hence $\mathsf{Hyb}_0(b)$ and $\mathsf{Hyb}_1(b)$ are identically distributed. □

Then, we define $\mathsf{Hyb}_2(b)$ as follows.

- $\mathsf{Hyb}_2(b)$: It is the same as $\mathsf{Hyb}_1(b)$ except that the challenger $\mathcal{C}$ generates the shares lazily using polynomial interpolation.

  Concretely, instead of invoking $\mathsf{Split}(\iota, s_\iota^{(b)}, \{\mathsf{mvk}_{i_1,i_2}\}_{i_1,i_2\in[n]})$ for every $\iota \in [M]$, $\mathcal{C}$ goes through the following process.

  1. For every $\iota \in [M]$, sample $\sigma_\iota \leftarrow \mathbb{Z}_q$ uniformly at random, and let $f_\iota(0) := s_\iota^{(b)} - \sigma_\iota$.
  2. For every $\iota \in [M]$, every $i_1, i_2 \in [n]$, and every $j \in \mathsf{mvk}_{i_2,i_1}$, sample $r_{i_1,j,\iota} \leftarrow \mathbb{Z}_q$, and set $|\psi_{i_1,j,\iota}\rangle := \mathsf{QFT}|r_{i_1,j,\iota}\rangle$. Let $\mathsf{vk}_{i_2,i_1,\iota} := \{(j, r_{i_1,j,\iota})\}_{j\in\mathsf{mvk}_{i_2,i_1}}$.
  3. For every $i_1 \in [n]$, randomly partition $\mathcal{J}_{i_1}^{\checkmark}$ into two sets $\mathcal{J}_{i_1}^{L}$ and $\mathcal{J}_{i_1}^{R}$ such that $|\mathcal{J}_{i_1}^{L}| = m'_L$ (and thus $|\mathcal{J}_{i_1}^{R}| = m'_R$). For every $j \in \mathcal{J}_{i_1}^{L}$, $\iota \in [M]$, sample $f_\iota((i_1-1)m+j) \leftarrow \mathbb{Z}_q$, and set $|\psi_{i_1,j,\iota}\rangle := |f_\iota((i_1-1)m+j)\rangle$. Let $\mathcal{K}$ record the nonzero evaluation points whose values are sampled independently. Initialize $\mathcal{K} := \bigcup_{i\in[n]}\{(i-1)m+j : j \in \mathcal{J}_i^{L}\}$. This set gains the R points of each of the first t quantum-share deliveries. Once t such deliveries have occurred, $|\mathcal{K}| = d$ and $\mathcal{K}$ remains fixed, including after deletions.
  4. When $\mathcal{A}$ corrupts $\mathcal{P}_{i_\tau^{(C)}}$ for some $\tau \in [1, \hat{\tau}]$ with $\varphi(i_\tau^{(C)}) = 1$, the challenger $\mathcal{C}$, for every $\iota \in [M]$ and every $j \in \mathcal{J}_{i_\tau^{(C)}}^{R}$, samples $f_\iota((i_\tau^{(C)}-1)m+j) \leftarrow \mathbb{Z}_q$, and sets $|\psi_{i_\tau^{(C)},j,\iota}\rangle := |f_\iota((i_\tau^{(C)}-1)m+j)\rangle$. Update $\mathcal{K} \leftarrow \mathcal{K} \cup \{(i_\tau^{(C)}-1)m+j : j \in \mathcal{J}_{i_\tau^{(C)}}^{R}\}$. For each $\iota \in [M]$, let $\mathsf{sh}_{i_\tau^{(C)},\iota} := (|\boldsymbol{\psi}_{i_\tau^{(C)},\iota}\rangle, \sigma_\iota)$. Then $\mathcal{C}$ sends
  $$(\{\mathsf{mvk}_{i_\tau^{(C)},i_2}\}_{i_2\in[n]}, \{\mathsf{vk}_{i_\tau^{(C)},i_2,\iota}\}_{i_2\in[n],\iota\in[M]}, \{\mathsf{sh}_{i_\tau^{(C)},\iota}\}_{\iota\in[M]})$$
  to $\mathcal{A}$.
  5. When $\mathcal{A}$ corrupts $\mathcal{P}_{i_\tau^{(C)}}$ for some $\tau \in (\hat{\tau}, n]$ with $\varphi(i_\tau^{(C)}) = 1$, for every $\iota \in [M]$ and $j \in \mathcal{J}_{i_\tau^{(C)}}^{R}$, interpolate $f_\iota((i_\tau^{(C)}-1)m+j)$ from $\{f_\iota(k)\}_{k\in\{0\}\cup\mathcal{K}}$ using Lemma 7, and set $|\psi_{i_\tau^{(C)},j,\iota}\rangle := |f_\iota((i_\tau^{(C)}-1)m+j)\rangle$. For each $\iota \in [M]$, let $\mathsf{sh}_{i_\tau^{(C)},\iota} := (|\boldsymbol{\psi}_{i_\tau^{(C)},\iota}\rangle, \sigma_\iota)$. $\mathcal{C}$ then sends
  $$(\{\mathsf{mvk}_{i_\tau^{(C)},i_2}\}_{i_2\in[n]}, \{\mathsf{vk}_{i_\tau^{(C)},i_2,\iota}\}_{i_2\in[n],\iota\in[M]}, \{\mathsf{sh}_{i_\tau^{(C)},\iota}\}_{\iota\in[M]})$$
  to $\mathcal{A}$.

6. When $\mathcal{A}$ corrupts $\mathcal{P}_{i_\tau^{(C)}}$ for some $\tau \in [n]$ with $\varphi(i_\tau^{(C)}) = 0$, the challenger sends

$$(\{\mathsf{mvk}_{i_\tau^{(C)},i_2}\}_{i_2\in[n]}, \{\mathsf{vk}_{i_\tau^{(C)},i_2,\iota}\}_{i_2\in[n],\iota\in[M]}, \{\sigma_\iota\}_{\iota\in[M]})$$

   to $\mathcal{A}$.

7. All remaining aspects of the experiment are identical to those in $\mathsf{Hyb}_1(b)$.

Lemma 29. $\mathsf{TD}(\mathsf{Hyb}_1(b), \mathsf{Hyb}_2(b)) = 0$.

Proof. Conditioned on $f_\iota(0)$, the values of a uniformly random polynomial of degree at most d at any at most d distinct nonzero points are independent and uniform, by Lemma 7. The sampled L and R values in $\mathsf{Hyb}_2(b)$ account for at most d such points. Once d values have been sampled, interpolation from $\{0\} \cup \mathcal{K}$ uniquely determines all remaining values. Sampling these values when needed therefore gives the same distribution, including under adaptive corruptions. □

We continue to define $\mathsf{Hyb}_3(b)$ as follows.

- $\mathsf{Hyb}_3(b)$: This game is the same as $\mathsf{Hyb}_2(b)$ except that $\mathcal{C}$ purifies the share-generation procedure.

  For every $i \in [n]$ and $\iota \in [M]$, use the registers

$$\mathsf{C}_{i,\iota} := (\mathsf{C}_{i,1,\iota}, \ldots, \mathsf{C}_{i,m,\iota}) \quad \text{and} \quad \mathsf{A}_{i,\iota} := (\mathsf{A}_{i,1,\iota}, \ldots, \mathsf{A}_{i,m,\iota}).$$

  When $\mathcal{P}_i$ is corrupted before deletion, the share delivered is $(\mathsf{A}_{i,\iota}, \sigma_\iota)$. The challenger retains all C registers and all A registers that have not been delivered. For every $\iota \in [M]$, define

$$\mathsf{C}_\iota^{\mathsf{base}} := \left(\mathsf{C}_{\lceil k/m\rceil,((k-1) \bmod m)+1,\iota}\right)_{k\in\mathcal{K}},$$

  with the registers listed in a fixed order. Here $\mathcal{K}$ evolves as in $\mathsf{Hyb}_2(b)$, and $\mathsf{C}_\iota^{\mathsf{base}}$ consists of the corresponding private data registers originally prepared as halves of fresh EPR pairs. The register $\mathsf{C}_{0,\iota}$ is kept separately. The joint states are prepared as follows.

  1. For every $\iota \in [M]$, sample $\sigma_\iota \leftarrow \mathbb{Z}_q$, set $f_\iota(0) := s_\iota^{(b)} - \sigma_\iota$, and prepare the private register $\mathsf{C}_{0,\iota}$ in $|f_\iota(0)\rangle$. For every $i \in [n]$ and $j \in [m]$, initialize $\mathsf{C}_{i,j,\iota}$ and $\mathsf{A}_{i,j,\iota}$ to $|0\rangle$.
  2. For every $\iota \in [M]$, every $i_1, i_2 \in [n]$, and every $j \in \mathsf{mvk}_{i_2,i_1}$, prepare the state

$$q^{-1/2} \sum_{x\in\mathbb{Z}_q} |x\rangle_{\mathsf{C}_{i_1,j,\iota}} |x\rangle_{\mathsf{A}_{i_1,j,\iota}} = q^{-1/2} \sum_{x\in\mathbb{Z}_q} \mathsf{QFT}^{-1}|x\rangle_{\mathsf{C}_{i_1,j,\iota}} \mathsf{QFT}|x\rangle_{\mathsf{A}_{i_1,j,\iota}}$$

     on the two registers $\mathsf{C}_{i_1,j,\iota}$ and $\mathsf{A}_{i_1,j,\iota}$. Next, measure $\mathsf{C}_{i_1,j,\iota}$ on the inverse-QFT basis, and let the outcome be $r_{i_1,j,\iota}$. Define $\mathsf{vk}_{i_2,i_1,\iota} := \{(j, r_{i_1,j,\iota})\}_{j\in\mathsf{mvk}_{i_2,i_1}}$.
  3. For every $i_1 \in [n]$, choose the subsets $\mathcal{J}_{i_1}^L, \mathcal{J}_{i_1}^R$ exactly as in $\mathsf{Hyb}_2(b)$, using the same partition for all ι. For every $\iota \in [M]$ and $j \in \mathcal{J}_{i_1}^L$, prepare the state

$$q^{-1/2} \sum_{x\in\mathbb{Z}_q} |x\rangle_{\mathsf{C}_{i_1,j,\iota}} \otimes |x\rangle_{\mathsf{A}_{i_1,j,\iota}}$$

     on the two registers $\mathsf{C}_{i_1,j,\iota}$ and $\mathsf{A}_{i_1,j,\iota}$.

4. When $\mathcal{A}$ corrupts $\mathcal{P}_{i_\tau^{(C)}}$ for some $\tau \in [1, \hat{\tau}]$ with $\varphi(i_\tau^{(C)}) = 1$, the challenger $\mathcal{C}$, for every $\iota \in [M]$ and every $j \in \mathcal{J}^R_{i_\tau^{(C)}}$, prepares the state

$$q^{-1/2} \sum_{x \in \mathbb{Z}_q} |x\rangle_{\mathsf{C}_{i_\tau^{(C)},j,\iota}} \otimes |x\rangle_{\mathsf{A}_{i_\tau^{(C)},j,\iota}}$$

on the two registers $\mathsf{C}_{i_\tau^{(C)},j,\iota}$ and $\mathsf{A}_{i_\tau^{(C)},j,\iota}$. For each $\iota \in [M]$, let $\mathsf{sh}_{i_\tau^{(C)},\iota} := (\mathsf{A}_{i_\tau^{(C)},\iota}, \sigma_\iota)$. The challenger $\mathcal{C}$ then gives

$$(\{\mathsf{mvk}_{i_\tau^{(C)},i_2}\}_{i_2 \in [n]}, \{\mathsf{vk}_{i_\tau^{(C)},i_2,\iota}\}_{i_2 \in [n], \iota \in [M]}, \{\mathsf{sh}_{i_\tau^{(C)},\iota}\}_{\iota \in [M]})$$

to $\mathcal{A}$.

5. When $\mathcal{A}$ corrupts $\mathcal{P}_{i_\tau^{(C)}}$ for some $\tau \in (\hat{\tau}, n]$ with $\varphi(i_\tau^{(C)}) = 1$, use the fixed source set $\{0\} \cup \mathcal{K}$. For every $\iota \in [M]$ and $j \in \mathcal{J}^R_{i_\tau^{(C)}}$, coherently interpolate from the following source registers, preserving their computational-basis labels:

$$\mathsf{C}_{i_\tau^{(C)},j,\iota} \leftarrow \mathsf{Interpolate}_{\mathbb{Z}_q,d}\big((i_\tau^{(C)} - 1)m + j,\ \{(0, \mathsf{C}_{0,\iota})\} \cup \{(k, \mathsf{C}_{\lceil k/m \rceil, ((k-1) \bmod m)+1, \iota})\}_{k \in \mathcal{K}}\big),$$

and then copy the content of $\mathsf{C}_{i_\tau^{(C)},j,\iota}$ to $\mathsf{A}_{i_\tau^{(C)},j,\iota}$ in the computational basis. For each $\iota \in [M]$, let $\mathsf{sh}_{i_\tau^{(C)},\iota} := (\mathsf{A}_{i_\tau^{(C)},\iota}, \sigma_\iota)$. The challenger $\mathcal{C}$ then gives

$$(\{\mathsf{mvk}_{i_\tau^{(C)},i_2}\}_{i_2 \in [n]}, \{\mathsf{vk}_{i_\tau^{(C)},i_2,\iota}\}_{i_2 \in [n], \iota \in [M]}, \{\mathsf{sh}_{i_\tau^{(C)},\iota}\}_{\iota \in [M]})$$

to $\mathcal{A}$.

6. When $\mathcal{A}$ corrupts $\mathcal{P}_{i_\tau^{(C)}}$ for some $\tau \in [n]$ with $\varphi(i_\tau^{(C)}) = 0$, the challenger sends

$$(\{\mathsf{mvk}_{i_\tau^{(C)},i_2}\}_{i_2 \in [n]}, \{\mathsf{vk}_{i_\tau^{(C)},i_2,\iota}\}_{i_2 \in [n], \iota \in [M]}, \{\sigma_\iota\}_{\iota \in [M]})$$

to $\mathcal{A}$.

7. All remaining aspects of the experiment are identical to those in $\mathsf{Hyb}_2(b)$.

Lemma 30. $\mathsf{TD}(\mathsf{Hyb}_2(b), \mathsf{Hyb}_3(b)) = 0$.

Proof. We compare the two experiments step by step.

- In Step 2 of $\mathsf{Hyb}_2(b)$, for every $\iota \in [M]$, every $i_1, i_2 \in [n]$, and every $j \in \mathsf{mvk}_{i_2,i_1}$, the challenger prepares a uniformly random QFT-basis state $|\psi_{i_1,j,\iota}\rangle$. In Step 2 of $\mathsf{Hyb}_3(b)$, the reduced state on the register $\mathsf{A}_{i_1,j,\iota}$ is distributed identically. Hence the adversary's view in these two steps is the same.

- The same reasoning applies to Steps 3 and 4, except that the relevant states are uniformly random computational-basis states rather than QFT-basis states. Thus, the adversary's view is again identical in the two hybrids.

- It remains to analyze Step 5. Consider measuring every freshly prepared data register $\mathsf{C}_{i,j,\iota}$ in the computational basis immediately after its EPR preparation. These measurements commute with all later interpolation operations, since the original registers are retained and used only as computational-basis controls, and with all operations on the adversary's registers. They can therefore be postponed until the end, where they do not affect the adversary's reduced state. With the measurements performed immediately, their outcomes are exactly the uniformly sampled values in $\mathsf{Hyb}_2(b)$, and every later share is interpolated from the same fixed set $\{0\} \cup \mathcal{K}$. Thus the two experiments give the same adversarial view.

This proves the claim. □

Before defining the next sequence of hybrids, we specify the additional projectors used in the proof. For $\mathcal{J} = \{j_1 < \cdots < j_h\} \subseteq [m]$, write $\mathsf{C}_{i,\iota}[\mathcal{J}] := (\mathsf{C}_{i,j_1,\iota}, \ldots, \mathsf{C}_{i,j_h,\iota})$ and define $\mathsf{A}_{i,\iota}[\mathcal{J}]$ analogously. We retain all prepared private data registers until the end, including the never-delivered $\mathsf{A}_{i,\iota}[\mathcal{J}_i^L]$ register for $\mathcal{P}_i$ with $\varphi(i) = 0$.

Let $\mathcal{B} := \{i \in \mathcal{CS} : \varphi(i) = 1\}$ denote the set of parties whose quantum shares have been delivered to $\mathcal{A}$. When an undeleted party $\mathcal{P}_i$, with $i = i_\tau^{(C)}$, is corrupted, we write the coherent data-generation map as

$$\mathbf{x} \in \mathbb{Z}_q^{|\mathcal{K}|},\ |\mathbf{x}\rangle_{\mathsf{C}_\iota^{\mathsf{base}}} |\mathbf{0}\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]} \mapsto |\mathbf{x}\rangle_{\mathsf{C}_\iota^{\mathsf{base}}} |\mathbf{L}_i \mathbf{x} + f_\iota(0)\mathbf{l}_i\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]},$$

where $\mathbf{L}_i$ and $\mathbf{l}_i$ are determined by the generation rule: if $\tau \le \widehat{\tau}$, each row of $\mathbf{L}_i$ selects the corresponding base coordinate and $\mathbf{l}_i = \mathbf{0}$; otherwise, the coordinates on $\mathcal{J}_i^L$ are copied in the same way, while those on $\mathcal{J}_i^R$ are interpolated from $\{0\} \cup \mathcal{K}$, with the corresponding rows of $\mathbf{L}_i$ containing the Lagrange coefficients at the points in $\mathcal{K}$ and the corresponding entries of $\mathbf{l}_i$ containing the coefficient at 0. With a slight abuse of notation, let $\mathbf{L}_\mathcal{B}$ and $\mathbf{l}_\mathcal{B}$ denote the vertical concatenations of $\mathbf{L}_i$ and $\mathbf{l}_i$, respectively, over $i \in \mathcal{B}$ in a fixed party order, so that the joint delivered data vector is $\mathbf{L}_\mathcal{B}\mathbf{x} + f_\iota(0)\mathbf{l}_\mathcal{B}$. For each base label $\mathbf{x}$, let $|\mathbf{w}_\iota(\mathbf{x})\rangle_{\mathsf{W}_\iota}$ denote the computational-basis state of all remaining private data registers under the current generation history, including any private register storing $f_\iota(0)$.

Put $\omega_q := e^{2\pi\sqrt{-1}/q}$. For $\mathbf{y} = (\mathbf{y}_i)_{i\in\mathcal{B}} \in (\mathbb{Z}_q^{m'})^\mathcal{B}$, define

$$|\mathsf{priv}_\iota(\mathbf{y})\rangle := q^{-|\mathcal{K}|/2} \sum_{\mathbf{x}\in\mathbb{Z}_q^{|\mathcal{K}|}} \omega_q^{-\mathbf{y}^\mathsf{T}(\mathbf{L}_\mathcal{B}\mathbf{x} + f_\iota(0)\mathbf{l}_\mathcal{B})} |\mathbf{x}\rangle_{\mathsf{C}_\iota^{\mathsf{base}}} |\mathbf{w}_\iota(\mathbf{x})\rangle_{\mathsf{W}_\iota}. \tag{9}$$

The vectors $\mathbf{y}$ are Fourier-expansion indices; defining this family performs no measurement and introduces no new physical registers.

For any $\mathcal{D} \subseteq \mathcal{B}$ with $|\mathcal{B} \setminus \mathcal{D}| \le t$, write $\mathsf{cert}_\mathcal{D} := \{\mathsf{cert}_{i,\iota}\}_{i\in\mathcal{D},\,\iota\in[M]}$ for its certificate family and define

$$\mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D}) := \big\{\mathbf{y} \in (\mathbb{Z}_q^{m'})^\mathcal{B} \mid \forall i \in \mathcal{D},\ \mathsf{HW}(\mathbf{y}_i - \mathsf{cert}_{i,\iota}[\mathcal{J}_i^{\checkmark}]) < m_L'/2\big\}.$$

Define

$$\mathbf{\Pi}^{\mathsf{glob}}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D}) := \sum_{\mathbf{y}\in\mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D})} |\mathsf{priv}_\iota(\mathbf{y})\rangle\langle\mathsf{priv}_\iota(\mathbf{y})|,$$

and set

$$\mathbf{\Pi}^{\mathsf{glob}}_{\mathcal{B},\mathcal{D}}(\mathsf{cert}_\mathcal{D}) := \bigotimes_{\iota=1}^{M} \mathbf{\Pi}^{\mathsf{glob}}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D}). \tag{10}$$

Lemma 31. *Suppose that $\mathcal{D} \subseteq \mathcal{B}$ and $|\mathcal{B}\backslash\mathcal{D}| \le t$. For every $\iota \in [M]$, the family $\{|\mathsf{priv}_\iota(\mathbf{y})\rangle\}_{\mathbf{y}\in\mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D})}$ is orthonormal. Consequently, the operators defined above are orthogonal projectors.*

Proof. Fix $\iota \in [M]$ and $\mathbf{y}, \mathbf{z} \in \mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D})$. Put $\mathbf{u} := \mathbf{y} - \mathbf{z}$. By Eqn (9),

$$\langle \mathsf{priv}_\iota(\mathbf{z})|\mathsf{priv}_\iota(\mathbf{y})\rangle = q^{-|\mathcal{K}|} \sum_{\mathbf{x}\in\mathbb{Z}_q^{|\mathcal{K}|}} \omega_q^{-\mathbf{u}^\mathsf{T}(\mathbf{L}_\mathcal{B}\mathbf{x}+f_\iota(0)\mathbf{l}_\mathcal{B})}.$$

For $\mathbf{y} = \mathbf{z}$, this equals 1. For $\mathbf{y} \neq \mathbf{z}$, it suffices to prove $\mathbf{L}_\mathcal{B}^\mathsf{T}\mathbf{u} \neq \mathbf{0}$.

For each $i \in \mathcal{D}$, both $\mathbf{y}_i$ and $\mathbf{z}_i$ are at Hamming distance less than $m'_L/2$ from the same certificate $\mathsf{cert}_{i,\iota}[\mathcal{J}_i^\checkmark]$. Thus $\mathsf{HW}(\mathbf{u}_i) < m'_L$ by the triangle inequality. For $i \in \mathcal{B} \setminus \mathcal{D}$, we use the bound $\mathsf{HW}(\mathbf{u}_i) \le m'$. Consequently,

$$\mathsf{HW}(\mathbf{u}) \le |\mathcal{B} \setminus \mathcal{D}|m' + |\mathcal{D}|m'_L \le nm'_L + tm'_R = d,$$

where the second inequality uses $|\mathcal{B}| \le n$, $|\mathcal{B} \setminus \mathcal{D}| \le t$, and $m' = m'_L + m'_R$.

If $|\mathcal{B}| \le t$, each row of $\mathbf{L}_\mathcal{B}$ selects a different base coordinate, so $\mathbf{L}_\mathcal{B}^\mathsf{T}\mathbf{u} \neq \mathbf{0}$ whenever $\mathbf{u} \neq \mathbf{0}$. Otherwise, $|\mathcal{K}| = d$. Since $\mathbf{u} \neq \mathbf{0}$ and $\mathsf{HW}(\mathbf{u}) \le d$, it suffices to show that every set of at most d rows of $\mathbf{L}_\mathcal{B}$ is linearly independent. This follows from its interpolation structure: after an invertible column transformation, its rows are $(\alpha, \alpha^2, \ldots, \alpha^d)$ at distinct nonzero evaluation points α, so the claim follows from the Vandermonde determinant. □

Define $\mathsf{Hyb}_{4,0}(b) := \mathsf{Hyb}_3(b)$. In each experiment, initialize $\mathcal{D} := \emptyset$ to record the parties whose certificates have passed the additional measurements. All interpolated R coordinates in these hybrids are computed from $\{0\} \cup \mathcal{K}$.

- $\mathsf{Hyb}_{4,\tau'}(b)$ for $\tau' \in [1, n]$:
  1. After $\mathcal{P}_{i_{\tau'}^{(D)}}$ deletes its share and produces the deletion certificate $\mathsf{cert}_{i_{\tau'}^{(D)}}$, if $|\mathcal{CS}| < n$ and Verify accepts for every uncorrupted party and every $\iota \in [M]$, let $\mathcal{D}^+ := \mathcal{D} \cup \{i_{\tau'}^{(D)}\}$. The challenger $\mathcal{C}$ immediately performs the two-outcome measurement
  $$\left\{\mathbf{\Pi}^{\mathsf{glob}}_{\mathcal{B},\mathcal{D}^+}(\mathsf{cert}_{\mathcal{D}^+}), \mathbf{I} - \mathbf{\Pi}^{\mathsf{glob}}_{\mathcal{B},\mathcal{D}^+}(\mathsf{cert}_{\mathcal{D}^+})\right\}$$
  on the private registers $\bigotimes_{\iota=1}^M(\mathsf{C}_\iota^{\mathsf{base}} \otimes \mathsf{W}_\iota)$. If the result is "reject", then the challenger outputs $\perp$ and aborts; otherwise, it updates $\mathcal{D} \leftarrow \mathcal{D}^+$. If this deletion does not occur or $|\mathcal{CS}| = n$, no additional measurement is performed.
  2. All remaining steps are the same as in $\mathsf{Hyb}_{4,\tau'-1}(b)$.

In addition, we define a sequence of simulators $\mathsf{Sim}_{4,\tau'}$ for $\tau' \in [1, n]$. These simulators are independent of the secret batch $\{s_\iota^{(b)}\}_{\iota\in[M]}$.

Define

$$\mathbf{\Pi}^{\mathsf{loc}}(\mathsf{cert}_{i,\iota}) := \sum_{\mathbf{z}\in\mathbb{Z}_q^{m'}:\ \mathsf{HW}(\mathbf{z}-\mathsf{cert}_{i,\iota}[\mathcal{J}_i^\checkmark])<m'_L/2} (\mathsf{QFT}^{-1})^{\otimes m'}|\mathbf{z}\rangle\langle\mathbf{z}|\mathsf{QFT}^{\otimes m'}.$$

This projector acts on $\mathsf{C}_{i,\iota}[\mathcal{J}_i^\checkmark]$; put $\mathbf{\Pi}^{\mathsf{loc}}(\mathsf{cert}_i) := \bigotimes_{\iota=1}^M \mathbf{\Pi}^{\mathsf{loc}}(\mathsf{cert}_{i,\iota})$.

- $\mathsf{Sim}_{4,\tau'}$ for $\tau' \in [1,n]$: This simulator is identical to $\mathsf{Hyb}_{4,\tau'}(b)$, except for the following changes. Whenever $\mathcal{A}$ corrupts an undeleted party $\mathcal{P}_i$, for every $\iota \in [M]$ and $j \in \mathcal{J}_i^R$, the challenger generates the corresponding data as the fresh EPR pair

$$q^{-1/2} \sum_{x \in \mathbb{Z}_q} |x\rangle_{\mathsf{C}_{i,j,\iota}} \otimes |x\rangle_{\mathsf{A}_{i,j,\iota}}.$$

  At every scheduled additional check for party i, it performs $\{\mathbf{\Pi}^{\mathsf{loc}}(\mathsf{cert}_i), \mathbf{I} - \mathbf{\Pi}^{\mathsf{loc}}(\mathsf{cert}_i)\}$, with the same rejection and update rules. It omits $f_\iota(0)$ and any private register storing it.

Let $\mathcal{COR}(\upsilon)$ denote the processing of the corruption query for $\mathcal{P}_{i_\upsilon^{(C)}}$, and let $\mathcal{DEL}(\upsilon')$ denote the processing of the corrupt-then-delete query for $\mathcal{P}_{i_{\upsilon'}^{(D)}}$, including any scheduled additional check. Honest deletions do not advance either counter. Corruptions after honest deletion advance the corruption counter but not the corrupt-then-delete counter. Every view includes the transcript and abort outcome.

For every fixed $\tau' \in [1,n]$, define $\overline{\mathsf{Hyb}}_{4,\tau'}(b)$ and $\overline{\mathsf{Sim}}_{4,\tau'}$ from $\mathsf{Hyb}_{4,\tau'}(b)$ and $\mathsf{Sim}_{4,\tau'}$, respectively, as follows. In each execution, let $\tilde{\tau}$ be the smallest corruption index satisfying $\eta(\tilde{\tau}) > \tau'$, if such an index exists. If $\tilde{\tau}$ exists, the experiment stops immediately before processing $\mathcal{COR}(\tilde{\tau})$; otherwise, it stops immediately after processing $\mathcal{COR}(n)$, if this event occurs. It outputs the adversary's view at that point. If the underlying experiment terminates without reaching either stopping event, it returns its terminating view, including any abort.

For $\tau' = n$, the first stopping condition is never satisfied. After $\mathcal{COR}(n)$, all parties have been corrupted, so there are no remaining calls to Verify and no further additional check is performed. The remaining adversarial view is therefore obtained from the stopped view by a common secret-independent channel, which acts as the identity on branches that have already terminated. By contractivity,

$$\mathsf{TD}(\mathsf{Hyb}_{4,n}(0), \mathsf{Hyb}_{4,n}(1)) \leq \mathsf{TD}(\overline{\mathsf{Hyb}}_{4,n}(0), \overline{\mathsf{Hyb}}_{4,n}(1)). \tag{11}$$

We use the following two lemmas.

Lemma 32. *For every fixed $\tau' \in [1,n]$ and $b \in \{0,1\}$, $\mathsf{TD}(\overline{\mathsf{Hyb}}_{4,\tau'}(b), \overline{\mathsf{Sim}}_{4,\tau'}) = 0$.*

Lemma 33. *Put $\varepsilon := 2M \exp(-\ell(m'_L)^2/(2m^2))$. For every $\tau' \in [1,n]$ and $b \in \{0,1\}$, $\mathsf{TD}(\mathsf{Hyb}_{4,\tau'-1}(b), \mathsf{Hyb}_{4,\tau'}(b)) \leq 2\sqrt{\varepsilon} + \varepsilon = 2^{-\Omega(\lambda)}$.*

Proof of Lemma 32. Fix $\tau' \in [1,n]$ and $b \in \{0,1\}$. We first describe the state of $\overline{\mathsf{Sim}}_{4,\tau'}$ and define its Fourier coefficient states. We then show that $\overline{\mathsf{Hyb}}_{4,\tau'}(b)$ has a corresponding expansion with exactly these coefficient states. Let R collect the side registers, including the adversary's registers, the checking registers, and any remaining purifying registers, but excluding the private data registers.

We first focus on $\overline{\mathsf{Sim}}_{4,\tau'}$. Immediately after any corruption or deletion event, let $\mathcal{B}$ and $\mathcal{D}$ denote the current sets. The simulator's current state admits an unnormalized purification $|\Psi^{\mathsf{Sim}}_{\mathcal{B},\mathcal{D}}\rangle$ of the form

$$|\Psi^{\mathsf{Sim}}_{\mathcal{B},\mathcal{D}}\rangle = \sum_{\{\mathbf{y}_\iota \in \mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D})\}_\iota} \left(\bigotimes_{\iota=1}^{M} \bigotimes_{i \in \mathcal{B}} (\mathsf{QFT}^{-1})^{\otimes m'} |\mathbf{y}_{i,\iota}\rangle_{\mathsf{C}_{i,\iota}[\mathcal{J}_i^{\checkmark}]} \right) |\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}, \tag{12}$$

where $\mathbf{y}_\iota = (\mathbf{y}_{i,\iota})_{i\in\mathcal{B}}$. We suppress the independent private EPR pairs corresponding to data registers that have not yet been delivered. For each $\{\mathbf{y}_\iota\}_\iota$, this expansion defines the *coefficient state* $|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}$ associated with these Fourier labels. The coefficient states may be unnormalized.

Initially, $\mathcal{B} = \mathcal{D} = \emptyset$. There is only the empty label, with coefficient state $|\chi_{\emptyset,\emptyset}(\emptyset)\rangle_{\mathsf{R}}$ containing the initial side information, so Eqn (12) holds.

For the induction step, we note that each operation on the side registers R can be represented by a Kraus operator $\mathbf{K}$, after including any necessary purifying registers in R. [3] This preserves the form of $|\Psi^{\mathsf{Sim}}_{\mathcal{B},\mathcal{D}}\rangle$, with each coefficient state $|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle$ updated to $\mathbf{K}|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle$. We now consider the corruption and deletion events.

- At $\mathcal{COR}(\tau)$, write $i := i^{(C)}_\tau$. If $\varphi(i) = 0$, then only classical information is disclosed, so the preceding observation applies and $\mathcal{B}, \mathcal{D}$ remain unchanged. Otherwise, the simulator delivers its quantum share. For each $\iota \in [M]$, the data EPR pairs satisfy

$$\begin{aligned} &q^{-m'/2} \sum_{\mathbf{x}\in\mathbb{Z}_q^{m'}} |\mathbf{x}\rangle_{\mathsf{C}_{i,\iota}[\mathcal{J}_i^{\checkmark}]} |\mathbf{x}\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]} \\ &\quad = q^{-m'/2} \sum_{\mathbf{z}_\iota\in\mathbb{Z}_q^{m'}} \left((\mathsf{QFT}^{-1})^{\otimes m'} |\mathbf{z}_\iota\rangle_{\mathsf{C}_{i,\iota}[\mathcal{J}_i^{\checkmark}]}\right) \otimes \left(\mathsf{QFT}^{\otimes m'} |\mathbf{z}_\iota\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]}\right). \end{aligned}$$

  Put $\mathcal{B}^+ := \mathcal{B} \cup \{i\}$ and $\mathsf{R}^+ := \mathsf{R} \otimes \bigotimes_{\iota=1}^M \mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]$. Let $\mathbf{K}$ represent the side operations on the current outcome branch, acting on R. The resulting joint state is

$$\begin{aligned} &q^{-Mm'/2} \sum_{\substack{\{\mathbf{y}_\iota\in\mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_{\mathcal{D}})\}_\iota \\ \{\mathbf{y}_\iota^+\in\mathbb{Z}_q^{m'}\}_\iota}} \left(\bigotimes_{\iota=1}^M \bigotimes_{j\in\mathcal{B}} (\mathsf{QFT}^{-1})^{\otimes m'} |\mathbf{y}_{j,\iota}\rangle_{\mathsf{C}_{j,\iota}[\mathcal{J}_j^{\checkmark}]}\right) \\ &\quad \otimes \left(\bigotimes_{\iota=1}^M (\mathsf{QFT}^{-1})^{\otimes m'} |\mathbf{y}_\iota^+\rangle_{\mathsf{C}_{i,\iota}[\mathcal{J}_i^{\checkmark}]}\right) \\ &\quad \otimes \mathbf{K}|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}} \otimes \bigotimes_{\iota=1}^M \mathsf{QFT}^{\otimes m'} |\mathbf{y}_\iota^+\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]}. \end{aligned}$$

  Define the new coefficient states by

$$\begin{aligned} &|\chi_{\mathcal{B}^+,\mathcal{D}}(\{(\mathbf{y}_\iota, \mathbf{y}_\iota^+)\}_\iota)\rangle_{\mathsf{R}^+} \\ &\quad := q^{-Mm'/2} \mathbf{K}|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}} \otimes \bigotimes_{\iota=1}^M \mathsf{QFT}^{\otimes m'} |\mathbf{y}_\iota^+\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]}. \end{aligned} \tag{13}$$

  Since $i \notin \mathcal{D}$, we have $(\mathbf{y}_\iota, \mathbf{y}_\iota^+) \in \mathcal{G}_{\mathcal{B}^+,\mathcal{D},\iota}(\mathsf{cert}_{\mathcal{D}})$. Therefore, the displayed joint state has precisely the required form $|\Psi^{\mathsf{Sim}}_{\mathcal{B}^+,\mathcal{D}}\rangle$ in Eqn (12).

[3] The reason we introduce purifying registers is that the adversary's operations need not be unitary. In particular, a measurement can be implemented by a unitary interaction with an auxiliary register that records the outcome. We retain such auxiliary registers, together with any environment that would otherwise be discarded, in R to purify these operations.

- At $\mathcal{DEL}(\upsilon')$, write $i := i^{(D)}_{\upsilon'}$. Let $\mathbf{K}$ represent the side operations on the current outcome branch since the preceding induction point, including certificate generation and verification using Verify, acting on R. If any call to Verify rejects, the experiment aborts.

  If no additional check is performed, $\mathcal{B}$ and $\mathcal{D}$ remain unchanged, and the induction claim holds with each coefficient state $|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}$ updated to $\mathbf{K}|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}$.

  Otherwise, put $\mathcal{D}^+ := \mathcal{D} \cup \{i\}$. The simulator's local projector keeps a Fourier term precisely when $\mathsf{HW}(\mathbf{y}_{i,\iota} - \mathsf{cert}_{i,\iota}[\mathcal{J}_i^{\checkmark}]) < m'_L/2$ for every ι. Since the previous certificate constraints already hold, the joint state after acceptance is

$$\sum_{\{\mathbf{y}_\iota \in \mathcal{G}_{\mathcal{B},\mathcal{D}^+,\iota}(\mathsf{cert}_{\mathcal{D}^+})\}_\iota} \left(\bigotimes_{\iota=1}^{M} \bigotimes_{j\in\mathcal{B}} (\mathsf{QFT}^{-1})^{\otimes m'} |\mathbf{y}_{j,\iota}\rangle_{\mathsf{C}_{j,\iota}[\mathcal{J}_j^{\checkmark}]} \right) \otimes \mathbf{K}|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}. \tag{14}$$

  For each $\{\mathbf{y}_\iota \in \mathcal{G}_{\mathcal{B},\mathcal{D}^+,\iota}(\mathsf{cert}_{\mathcal{D}^+})\}_\iota$, define the coefficient state by

$$|\chi_{\mathcal{B},\mathcal{D}^+}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}} := \mathbf{K}|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}.$$

  The displayed joint state therefore has precisely the required form $|\Psi^{\mathsf{Sim}}_{\mathcal{B},\mathcal{D}^+}\rangle$ in Eqn (12).

We now consider $\overline{\mathsf{Hyb}}_{4,\tau'}(b)$. Fix the common classical setup, including the masks σ_ι, and use the same purifications of the side operations as in the simulator. We prove by induction over the corruption and deletion events that every sequence of events and outcomes occurs with the same probability in the two experiments, including the outcomes of the additional checks. As long as the experiments have not aborted, the hybrid's current state admits an unnormalized purification of the form

$$|\Psi^{\mathsf{Hyb}}_{\mathcal{B},\mathcal{D}}\rangle = \sum_{\{\mathbf{y}_\iota \in \mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_{\mathcal{D}})\}_\iota} \left(\bigotimes_{\iota=1}^{M} |\mathsf{priv}_\iota(\mathbf{y}_\iota)\rangle \right) \otimes |\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}, \tag{15}$$

where each $|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}$ is exactly the coefficient state defined in the simulator for the same classical setup and the same events and outcomes.

We first prove that the family $\{|\mathsf{priv}_\iota(\mathbf{y})\rangle\}_{\mathbf{y}\in\mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_{\mathcal{D}})}$ is orthonormal. By Lemma 31, we only need to prove $|\mathcal{B} \setminus \mathcal{D}| \le t$. This can be proved as follows. When $|\mathcal{D}| < \tau'$, we have $\mathcal{D} = \{i \in \mathcal{DS} | \varphi(i) = 1\}$. Since $\mathcal{B} = \{i \in \mathcal{CS} | \varphi(i) = 1\}$, we have $\mathcal{B}\backslash\mathcal{D} = \{i \in \mathcal{CS}\backslash\mathcal{DS} | \varphi(i) = 1\}$. So $|\mathcal{B}\backslash\mathcal{D}| \le |\mathcal{CS}\backslash\mathcal{DS}| \le t$. When $|\mathcal{D}| = \tau'$, by the definition of $\overline{\mathsf{Hyb}}_{4,\tau'}(b)$, every corruption $\mathcal{COR}(\tau)$ in this game satisfies $\eta(\tau) \le \tau'$, which means $|\mathcal{B}| \le t + \tau'$. So we also have $|\mathcal{B} \setminus \mathcal{D}| \le t$.

Consequently, once Eqn (15) is established, orthonormality of the private states implies equality of the unnormalized adversarial states for the same events and outcomes, provided the experiments have not aborted. To conclude equality of the experiments' outputs, we must also account for rejection of an additional check.

Initially, $\mathcal{B} = \mathcal{D} = \emptyset$. For each ι, $|\mathsf{priv}_\iota(\emptyset)\rangle$ contains the EPR pairs on $\mathsf{C}_{i,\iota}[\mathcal{J}_i^L] \otimes \mathsf{A}_{i,\iota}[\mathcal{J}_i^L]$ for all $i \in [n]$, together with any private register storing $f_\iota(0)$. The side registers R contain the adversary's registers, the checking registers, and the remaining classical and purifying registers. Their initial state is exactly the simulator's initial coefficient state $|\chi_{\emptyset,\emptyset}(\emptyset)\rangle_{\mathsf{R}}$. Thus Eqn (15) holds. We now consider corruption and deletion.

- At $\mathcal{COR}(\tau)$, write $i := i_\tau^{(C)}$. If $\varphi(i) = 0$, only classical information is disclosed, so the common side operations preserve the claimed expansion. Otherwise, put $\mathcal{B}^+ := \mathcal{B} \cup \{i\}$ and $\mathsf{R}^+ := \mathsf{R} \otimes \bigotimes_{\iota=1}^{M} \mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]$. Let $\mathbf{K}$ denote the same side operations as in the simulator, acting on the old R up to the moment before data delivery. (Such a common $\mathbf{K}$ exists because, by the induction hypothesis, the adversary's views in the two experiments are identical up to this point.)

  For $\tau \le \widehat{\tau}$, the desired identity follows from the same EPR expansion as in the simulator. We now consider $\tau > \widehat{\tau}$, when the base registers remain unchanged. For each ι, let $\mathbf{Q}_i$ denote the hybrid's data-generation and delivery operation. Write

$$\mathsf{W}_\iota = \mathsf{W}_\iota^- \otimes \mathsf{A}_{i,\iota}[\mathcal{J}_i^L], \qquad \mathsf{W}_\iota^+ := \mathsf{W}_\iota^- \otimes \mathsf{C}_{i,\iota}[\mathcal{J}_i^R].$$

  Let $\mathbf{w}_\iota^-(\mathbf{x})$ be the restriction of $\mathbf{w}_\iota(\mathbf{x})$ to W_ι^-, and let $\mathbf{L}_i^R, \mathbf{l}_i^R$ be the rows and entries of $\mathbf{L}_i, \mathbf{l}_i$ corresponding to $\mathcal{J}_i^R$. Define

$$|\mathbf{w}_\iota^+(\mathbf{x})\rangle_{\mathsf{W}_\iota^+} := |\mathbf{w}_\iota^-(\mathbf{x})\rangle_{\mathsf{W}_\iota^-} \otimes |\mathbf{L}_i^R\mathbf{x} + f_\iota(0)\mathbf{l}_i^R\rangle_{\mathsf{C}_{i,\iota}[\mathcal{J}_i^R]}.$$

  Expanding the input private state, we obtain

$$\begin{aligned}
&\mathbf{Q}_i|\mathsf{priv}_\iota(\mathbf{y}_\iota)\rangle \\
&= q^{-|\mathcal{K}|/2} \sum_{\mathbf{x}\in\mathbb{Z}_q^{|\mathcal{K}|}} \omega_q^{-\mathbf{y}_\iota^{\mathsf{T}}(\mathbf{L}_\mathcal{B}\mathbf{x}+f_\iota(0)\mathbf{l}_\mathcal{B})} \mathbf{Q}_i \left(|\mathbf{x}\rangle_{\mathsf{C}_\iota^{\mathsf{base}}} |\mathbf{w}_\iota(\mathbf{x})\rangle_{\mathsf{W}_\iota} \right) \\
&= q^{-|\mathcal{K}|/2} \sum_{\mathbf{x}\in\mathbb{Z}_q^{|\mathcal{K}|}} \omega_q^{-\mathbf{y}_\iota^{\mathsf{T}}(\mathbf{L}_\mathcal{B}\mathbf{x}+f_\iota(0)\mathbf{l}_\mathcal{B})} |\mathbf{x}\rangle_{\mathsf{C}_\iota^{\mathsf{base}}} |\mathbf{w}_\iota^-(\mathbf{x})\rangle_{\mathsf{W}_\iota^-} \\
&\qquad \otimes |\mathbf{L}_i^R\mathbf{x} + f_\iota(0)\mathbf{l}_i^R\rangle_{\mathsf{C}_{i,\iota}[\mathcal{J}_i^R]} \otimes |\mathbf{L}_i\mathbf{x} + f_\iota(0)\mathbf{l}_i\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]} \\
&= q^{-(|\mathcal{K}|+m')/2} \sum_{\substack{\mathbf{x}\in\mathbb{Z}_q^{|\mathcal{K}|} \\ \mathbf{y}_\iota^+\in\mathbb{Z}_q^{m'}}} \omega_q^{-\mathbf{y}_\iota^{\mathsf{T}}(\mathbf{L}_\mathcal{B}\mathbf{x}+f_\iota(0)\mathbf{l}_\mathcal{B})} \omega_q^{-(\mathbf{y}_\iota^+)^{\mathsf{T}}(\mathbf{L}_i\mathbf{x}+f_\iota(0)\mathbf{l}_i)} \\
&\qquad \cdot |\mathbf{x}\rangle_{\mathsf{C}_\iota^{\mathsf{base}}} |\mathbf{w}_\iota^+(\mathbf{x})\rangle_{\mathsf{W}_\iota^+} \otimes \mathsf{QFT}^{\otimes m'} |\mathbf{y}_\iota^+\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]} \\
&= q^{-m'/2} \sum_{\mathbf{y}_\iota^+\in\mathbb{Z}_q^{m'}} |\mathsf{priv}_\iota(\mathbf{y}_\iota, \mathbf{y}_\iota^+)\rangle \otimes \mathsf{QFT}^{\otimes m'} |\mathbf{y}_\iota^+\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]}.
\end{aligned} \tag{16}$$

  The private vectors in the last line use the updated registers $\mathsf{C}_\iota^{\mathsf{base}} \otimes \mathsf{W}_\iota^+$.

  Here $\mathcal{K}$, the private registers, and the affine expressions refer to the generation history immediately after this delivery. The joint state is therefore

$$\begin{aligned}
q^{-Mm'/2} \sum_{\substack{\{\mathbf{y}_\iota\in\mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D})\}_\iota \\ \{\mathbf{y}_\iota^+\in\mathbb{Z}_q^{m'}\}_\iota}} &\left(\bigotimes_{\iota=1}^{M} |\mathsf{priv}_\iota(\mathbf{y}_\iota, \mathbf{y}_\iota^+)\rangle \right) \\
&\otimes \mathbf{K}|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}} \otimes \bigotimes_{\iota=1}^{M} \mathsf{QFT}^{\otimes m'} |\mathbf{y}_\iota^+\rangle_{\mathsf{A}_{i,\iota}[\mathcal{J}_i^{\checkmark}]}.
\end{aligned}$$

  The side factor, including $q^{-Mm'/2}$, is exactly the simulator's coefficient state on R^+ for the combined labels $\{(\mathbf{y}_\iota, \mathbf{y}_\iota^+)\}_\iota$, by Eqn (13). Since $i \notin \mathcal{D}$, the combined labels range over exactly $\mathcal{G}_{\mathcal{B}^+,\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D})$. Thus the displayed state has the required form $|\Psi_{\mathcal{B}^+,\mathcal{D}}^{\mathsf{Hyb}}\rangle$.

- At $\mathcal{DEL}(\upsilon')$, write $i := i_{\upsilon'}^{(D)}$. Let $\mathbf{K}$ denote the same side operations as in the simulator, including certificate generation and verification using Verify, acting on R. If any call to Verify rejects, both experiments abort. If no additional check is performed, the coefficient states are updated by the same $\mathbf{K}$, so the induction claim continues to hold.

  Otherwise, put $\mathcal{D}^+ := \mathcal{D} \cup \{i\}$. The private family is unchanged at this check and is orthonormal on the old allowed set. Since $\mathcal{G}_{\mathcal{B},\mathcal{D}^+,\iota}(\mathsf{cert}_{\mathcal{D}^+}) \subseteq \mathcal{G}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_{\mathcal{D}})$, the global projector keeps exactly the terms satisfying the new certificate constraints. The accepting joint state is

$$\sum_{\{\mathbf{y}_\iota \in \mathcal{G}_{\mathcal{B},\mathcal{D}^+,\iota}(\mathsf{cert}_{\mathcal{D}^+})\}_\iota} \left(\bigotimes_{\iota=1}^{M} |\mathsf{priv}_\iota(\mathbf{y}_\iota)\rangle \right) \otimes \mathbf{K} |\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}. \tag{17}$$

  For each retained family, the coefficient state is $|\chi_{\mathcal{B},\mathcal{D}^+}(\{\mathbf{y}_\iota\}_\iota)\rangle_{\mathsf{R}}$ already defined in the simulator after Eqn (14). Thus acceptance gives the required state $|\Psi^{\mathsf{Hyb}}_{\mathcal{B},\mathcal{D}^+}\rangle$.

  By orthonormality, acceptance gives the same unnormalized reduced state on R as in the simulator. Since rejection uses the complementary projectors, the same coefficient-state correspondence implies that the unnormalized reduced state on R after rejection is also identical to that in the simulator.

This completes the induction. □

Next we prove Lemma 33.

Proof of Lemma 33. Fix $\tau' \in [1, n]$ and $b \in \{0, 1\}$. The two hybrids differ only in the additional global measurement at $\mathcal{DEL}(\tau')$, performed after all calls to Verify accept and only if $|\mathcal{CS}| < n$. Let p_{bad} be the probability that $\mathsf{Hyb}_{4,\tau'}(b)$ reaches this check and its global projector rejects. By the Gentle Measurement Lemma (Lemma 4), it suffices to prove $p_{\mathsf{bad}} \leq \varepsilon$, where $\varepsilon := 2M \exp(-\ell(m'_L)^2/(2m^2))$. By the proof of Lemma 32, p_{bad} equals to the rejection probability of the local projector $\mathbf{\Pi}^{\mathsf{loc}}(\mathsf{cert}_{i_{\tau'}^{(D)}})$ in $\overline{\mathsf{Sim}}_{4,\tau'}$.

To this end, we define the following auxiliary experiment.

- $\widetilde{\mathsf{Sim}}_{4,\tau'}$ for $\tau' \in [1, n]$: This experiment is identical to $\mathsf{Sim}_{4,\tau'}$, except that the challenger $\mathcal{C}$ prepares all maximally entangled states at the beginning, while generating mvk_{i_1,i_2} and $\{\mathsf{vk}_{i_1,i_2,\iota}\}_{\iota\in[M]}$ lazily for each $i_1, i_2 \in [n]$.

  There are two situations in which mvk_{i_1,i_2} and $\{\mathsf{vk}_{i_1,i_2,\iota}\}_{\iota\in[M]}$ must be generated:

  - A party $\mathcal{P}_{i_1}$ is corrupted, in which case $\mathcal{C}$ must provide mvk_{i_1,i_2} and $\{\mathsf{vk}_{i_1,i_2,\iota}\}_{\iota\in[M]}$ for $\mathcal{A}$.
  - A corrupted party $\mathcal{P}_{i_2}$ deletes its share, in which case $\mathcal{C}$ must use mvk_{i_1,i_2} and $\{\mathsf{vk}_{i_1,i_2,\iota}\}_{\iota\in[M]}$ to verify the deletion certificate of $\mathcal{P}_{i_2}$.

  More concretely, $\mathcal{C}$ proceeds as follows.

  1. At the beginning of the experiment, for every $i \in [n]$, $j \in [m]$, and $\iota \in [M]$, the challenger $\mathcal{C}$ prepares the state

$$q^{-1/2} \sum_{x\in\mathbb{Z}_q} |x\rangle_{\mathsf{C}_{i,j,\iota}} \otimes |x\rangle_{\mathsf{A}_{i,j,\iota}} = q^{-1/2} \sum_{x\in\mathbb{Z}_q} \mathsf{QFT}^{-1}|x\rangle_{\mathsf{C}_{i,j,\iota}} \otimes \mathsf{QFT}|x\rangle_{\mathsf{A}_{i,j,\iota}}$$

on the two registers $\mathsf{C}_{i,j,\iota}$ and $\mathsf{A}_{i,j,\iota}$. For every $\iota \in [M]$, $\mathcal{C}$ also samples $\sigma_\iota \leftarrow \mathbb{Z}_q$ uniformly at random.

2. Whenever $\mathcal{A}$ requests deletion of the share of an honest party $\mathcal{P}_i$, the challenger $\mathcal{C}$ performs the honest deletion procedure by measuring $\mathsf{A}_{i,\iota}$ in the QFT basis for every $\iota \in [M]$, and returns the measurement outcomes to $\mathcal{A}$.
3. When $\mathcal{A}$ deletes the share of $\mathcal{P}_{i_{\varsigma'}^{(D)}}$ for some $\varsigma' \in [n]$, suppose the deletion certificate is $\{\mathsf{cert}_{i_{\varsigma'}^{(D)},\iota}\}_{\iota\in[M]}$. Let $\varsigma^* \in [n]$ be the number of the currently corrupted party, i.e., exactly the parties $\mathcal{P}_{i_1^{(C)}}, \ldots, \mathcal{P}_{i_{\varsigma^*}^{(C)}}$ have been corrupted at this point in the experiment. The challenger $\mathcal{C}$ must then generate $\mathsf{mvk}_{i_1,i_{\varsigma'}^{(D)}}$ and $\{\mathsf{vk}_{i_1,i_{\varsigma'}^{(D)},\iota}\}_{\iota\in[M]}$ for every party $\mathcal{P}_{i_1}$ that remains honest at this point.
   To this end, define
   $$\overline{\mathcal{CS}}_{\varsigma^*} := [n]\backslash\{i_\upsilon^{(C)}\}_{\upsilon\in[\varsigma^*]}$$
   to be the set of parties that are still honest after the first ς^* corruptions. $\mathcal{C}$ samples $\{\mathsf{mvk}_{i_1,i_{\varsigma'}^{(D)}}\}_{i_1\in\overline{\mathcal{CS}}_{\varsigma^*}}$ as $n-\varsigma^*$ disjoint subsets of $[m]\backslash\bigcup_{\upsilon\in[1,\varsigma^*]}\mathsf{mvk}_{i_\upsilon^{(C)},i_{\varsigma'}^{(D)}}$, each of which has size ℓ. Then, for every $i_1 \in \overline{\mathcal{CS}}_{\varsigma^*}$ and every $\iota \in [M]$, $\mathcal{C}$ measures $\mathsf{C}_{i_{\varsigma'}^{(D)},\iota}[\mathsf{mvk}_{i_1,i_{\varsigma'}^{(D)}}]$ on the inverse QFT basis to get $\mathsf{vk}_{i_1,i_{\varsigma'}^{(D)},\iota}$, and checks whether
   $$\mathsf{Verify}(i_1, i_{\varsigma'}^{(D)}, \iota, \mathsf{mvk}_{i_1,i_{\varsigma'}^{(D)}}, \mathsf{vk}_{i_1,i_{\varsigma'}^{(D)},\iota}, \mathsf{cert}_{i_{\varsigma'}^{(D)},\iota}) = \top.$$
   If any call to Verify rejects, the experiment aborts. Otherwise, if $\varsigma' \le \tau'$ and $\varsigma^* < n$, it performs the same additional local check as $\mathsf{Sim}_{4,\tau'}$, with the same update and abort rules.
4. When $\mathcal{A}$ corrupts $\mathcal{P}_{i_\varsigma^{(C)}}$ for some $\varsigma \in [n]$, let $\varsigma' \in [0,n]$ denote the number of corrupted parties that have deleted their shares before this corruption. At this point, the challenger $\mathcal{C}$ must generate $\mathsf{mvk}_{i_\varsigma^{(C)},i_2}$ and $\{\mathsf{vk}_{i_\varsigma^{(C)},i_2,\iota}\}_{\iota\in[M]}$ for every $i_2 \in [n]$.
   For every $i_2 \in \{i_{\upsilon'}^{(D)}\}_{\upsilon'\in[\varsigma']}$, the values $\mathsf{mvk}_{i_\varsigma^{(C)},i_2}$ and $\{\mathsf{vk}_{i_\varsigma^{(C)},i_2,\iota}\}_{\iota\in[M]}$ have already been generated in Step 3, namely when $\mathcal{P}_{i_2}$ deleted its share. Let
   $$\overline{\mathcal{DS}}_{\varsigma'} := [n]\backslash\{i_{\upsilon'}^{(D)}\}_{\upsilon'\in[\varsigma']}$$
   denote the set of parties whose shares have not previously been deleted while corrupted. This set may include honestly deleted parties; for those parties, the inverse-QFT measurements below return the corresponding entries of their previously issued certificates. It therefore remains to generate these values only for parties $i_2 \in \overline{\mathcal{DS}}_{\varsigma'}$.
   For each such i_2, $\mathcal{C}$ samples $\mathsf{mvk}_{i_\varsigma^{(C)},i_2}$ as a uniformly random subset of
   $$[m]\backslash\left(\bigcup_{\upsilon=1}^{\varsigma-1}\mathsf{mvk}_{i_\upsilon^{(C)},i_2}\right)$$
   of size ℓ. Then, for every $i_2 \in \overline{\mathcal{DS}}_{\varsigma'}$ and every $\iota \in [M]$, the challenger measures $\mathsf{C}_{i_2,\iota}[\mathsf{mvk}_{i_\varsigma^{(C)},i_2}]$ in the inverse-QFT basis to obtain $\mathsf{vk}_{i_\varsigma^{(C)},i_2,\iota}$. Finally, $\mathcal{C}$ sends $\{\mathsf{mvk}_{i_\varsigma^{(C)},i}\}_{i\in[n]}$ and $\{\mathsf{vk}_{i_\varsigma^{(C)},i,\iota}\}_{i\in[n],\iota\in[M]}$ to $\mathcal{A}$. If $\varphi(i_\varsigma^{(C)}) = 1$, $\mathcal{C}$ additionally sends $\{(\mathsf{A}_{i_\varsigma^{(C)},\iota}, \sigma_\iota)\}_{\iota\in[M]}$. Otherwise, $\mathcal{C}$ sends $\{\sigma_\iota\}_{\iota\in[M]}$.

We next show that the view of $\mathcal{A}$ in $\widetilde{\mathsf{Sim}}_{4,\tau'}$ is identical to its view in $\mathsf{Sim}_{4,\tau'}$.

- In Step 2, when $\mathcal{A}$ requests deletion of the share of an honest party $\mathcal{P}_i$, let $\varsigma^* \in [0,n]$ be the number of the currently corrupted party.

  We first claim that, for every $\upsilon \in [\varsigma^*]$, $\mathsf{cert}_{i,\iota}[\mathsf{mvk}_{i_\upsilon^{(C)},i}] = \mathsf{vk}_{i_\upsilon^{(C)},i,\iota}$. Indeed, for each such υ, the set $\mathsf{mvk}_{i_\upsilon^{(C)},i}$ and the values $\{\mathsf{vk}_{i_\upsilon^{(C)},i,\iota}\}_{\iota\in[M]}$ have already been determined. By construction, $\mathsf{vk}_{i_\upsilon^{(C)},i,\iota}$ is obtained by measuring $\mathsf{C}_{i,\iota}[\mathsf{mvk}_{i_\upsilon^{(C)},i}]$ on the inverse QFT basis. Therefore, when $\mathsf{A}_{i,\iota}[\mathsf{mvk}_{i_\upsilon^{(C)},i}]$ is measured on the QFT basis to obtain $\mathsf{cert}_{i,\iota}[\mathsf{mvk}_{i_\upsilon^{(C)},i}]$ the outcome is exactly $\mathsf{vk}_{i_\upsilon^{(C)},i,\iota}$, by the entanglement between the two registers. Consequently, the corresponding verification procedure accepts with certainty.

  Next, for every $j \in [m]\setminus\bigcup_{\upsilon=1}^{\varsigma^*}\mathsf{mvk}_{i_\upsilon^{(C)},i}$, the value $\mathsf{cert}_{i,\iota}[j]$ is uniformly random, since the register $\mathsf{A}_{i,j,\iota}$ remains a uniformly random state at that point.

  It follows that the distribution of $\mathsf{cert}_{i,\iota}$ is exactly the same as in $\mathsf{Sim}_{4,\tau'}$.

- In Steps 3 and 4, the only difference is that, for each $i_1, i_2 \in [n]$, the values mvk_{i_1,i_2} and $\{\mathsf{vk}_{i_1,i_2,\iota}\}_{\iota\in[M]}$ are generated only when they are first needed. All measurements of the C registers use the inverse-QFT basis, and the additional local projectors are diagonal in this basis. The checking subsets for a party are fully generated before its additional check. Thus deferring these measurements and sampling each unused subset when first needed preserves the adversary's view and the probabilities of the outcomes of the additional checks.

Thus, $\widetilde{\mathsf{Sim}}_{4,\tau'}$ and $\mathsf{Sim}_{4,\tau'}$ have the same adversarial view and the same probabilities for the outcomes of the additional checks. It remains to bound the probability that all calls to Verify accept and the local projector rejects at $\mathcal{DEL}(\tau')$ in $\widetilde{\mathsf{Sim}}_{4,\tau'}$.

For the analysis, fix the preceding events and outcomes and the certificate at $\mathcal{DEL}(\tau')$, before running Verify, with $\tau^* := |\mathcal{CS}| < n$. Here τ^* counts all corrupted parties, including those corrupted after honest deletion. Before sampling the remaining checking subsets for $\mathcal{P}_{i_{\tau'}^{(D)}}$, conceptually measure $\mathsf{C}_{i_{\tau'}^{(D)},\iota}$ in the inverse-QFT basis for each $\iota \in [M]$, and let $\mathbf{y}_{i_{\tau'}^{(D)},\iota} \in \mathbb{Z}_q^m$ denote the outcome. The challenger does not actually perform this full measurement; it is used only for the analysis.

Define

$$\mathcal{J}^*_{i_{\tau'}^{(D)}} := [m] \setminus \bigcup_{\upsilon\in[\tau^*]} \mathsf{mvk}_{i_\upsilon,i_{\tau'}^{(D)}}$$

and

$$\mathcal{J}^{*\times}_{i_{\tau'}^{(D)}} := \bigcup_{i_1\in\overline{\mathcal{CS}}_{\tau^*}} \mathsf{mvk}_{i_1,i_{\tau'}^{(D)}} = \mathcal{J}^*_{i_{\tau'}^{(D)}} \setminus \mathcal{J}^{\checkmark}_{i_{\tau'}^{(D)}},$$

where $\mathcal{J}^{\checkmark}_{i_{\tau'}^{(D)}}$ is defined at Eqn (6). Since the party being deleted has not previously been deleted, its remaining checking subsets have not been used by Verify or for an additional check. Checking subsets for different parties' shares are sampled independently. Consequently, $\mathcal{J}^{*\times}_{i_{\tau'}^{(D)}}$ is a uniformly random subset of $\mathcal{J}^*_{i_{\tau'}^{(D)}}$ of size $(n-\tau^*)\ell$, even conditioned on the conceptual outcomes. For the fixed preceding events, outcomes, and certificate, the probability that all calls to Verify accept and

the local projector rejects is at most

$$\Pr_{\substack{\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}},\{\mathbf{y}_{i^{(D)}_{\tau'},\iota}\}_{\iota\in[M]}}}\Bigg[\exists \iota\in[M],\Big(\mathsf{HW}(\mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{\checkmark}_{i^{(D)}_{\tau'}}]-\mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{\checkmark}_{i^{(D)}_{\tau'}}])\geq m'_L/2 \\ \wedge\ \mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}]=\mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}]\Big)\Bigg],$$

Since $\mathcal{J}^{\checkmark}_{i^{(D)}_{\tau'}}\subseteq\mathcal{J}^{*}_{i^{(D)}_{\tau'}}$, the probability above is no more than

$$\Pr_{\substack{\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}},\{\mathbf{y}_{i^{(D)}_{\tau'},\iota}\}_{\iota\in[M]}}}\Bigg[\exists \iota\in[M],\Big(\mathsf{HW}(\mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*}_{i^{(D)}_{\tau'}}]-\mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*}_{i^{(D)}_{\tau'}}])\geq m'_L/2 \\ \wedge\ \ \mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}]=\mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}]\Big)\Bigg],$$

Applying the union bound, this is at most

$$\sum_{\iota=1}^{M}\Pr_{\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}},\mathbf{y}_{i^{(D)}_{\tau'},\iota}}\Bigg[\mathsf{HW}(\mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*}_{i^{(D)}_{\tau'}}]-\mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*}_{i^{(D)}_{\tau'}}])\geq m'_L/2 \\ \wedge\ \mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}]=\mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}]\Bigg].$$

By Lemma 34, each summand is upper bounded by $2\exp(-\frac{\ell {m'_L}^2}{2m^2})$. Averaging over the preceding events, outcomes, and certificates gives $p_{\mathsf{bad}}\leq\varepsilon$.

After this check, neither hybrid performs further additional checks, and both continue interpolating from $\{0\}\cup\mathcal{K}$; the update of $\mathcal{D}$ is not used thereafter. Apply the Gentle Measurement Lemma to the joint state including the classical events and outcomes, treating the additional measurement as the identity whenever no check is reached. Accounting for abort on rejection and applying contractivity gives

$$\mathsf{TD}(\mathsf{Hyb}_{4,\tau'-1}(b),\mathsf{Hyb}_{4,\tau'}(b))\leq 2\sqrt{p_{\mathsf{bad}}}+p_{\mathsf{bad}}\leq 2\sqrt{\varepsilon}+\varepsilon=2^{-\Omega(\lambda)},$$

where the last equality uses $M=\mathrm{poly}(\lambda)$ and $\ell(m'_L)^2/m^2=\Omega(\lambda)$. □

Lemma 34. *Let $\mathcal{J}^{*}_{i^{(D)}_{\tau'}}:=[m]\setminus\bigcup_{\upsilon\in[\tau^*]}\mathsf{mvk}_{i_\upsilon,i^{(D)}_{\tau'}}$. Assume $\tau^*<n$ and fix $\iota\in[M]$, the preceding events and outcomes, and any certificate $\mathsf{cert}_{i^{(D)}_{\tau'},\iota}\in\mathbb{Z}_q^m$ before running Verify. Let $\mathbf{y}_{i^{(D)}_{\tau'},\iota}$ denote the outcome of conceptually measuring $\mathsf{C}_{i^{(D)}_{\tau'},\iota}$ in the inverse-QFT basis. Independently of this outcome and the certificate, sample $\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}$ uniformly among the $(n-\tau^*)\ell$-subsets of $\mathcal{J}^{*}_{i^{(D)}_{\tau'}}$. Then*

$$\Pr_{\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}},\mathbf{y}_{i^{(D)}_{\tau'},\iota}}\Bigg[\mathsf{HW}\big(\mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*}_{i^{(D)}_{\tau'}}]-\mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*}_{i^{(D)}_{\tau'}}]\big)\geq m'_L/2 \\ \wedge\ \mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}]=\mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}]\Bigg]\leq 2\exp\left(-\frac{\ell {m'_L}^2}{2m^2}\right).$$

Proof. Fix any outcome $\mathbf{y}_{i^{(D)}_{\tau'},\iota}$ such that

$$\mathsf{HW}\big(\mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^*_{i^{(D)}_{\tau'}}] - \mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^*_{i^{(D)}_{\tau'}}]\big) \geq m'_L/2.$$

For every $j \in \mathcal{J}^*_{i^{(D)}_{\tau'}}$, define

$$\zeta_j := \begin{cases} 1, & \text{if } \mathbf{y}_{i^{(D)}_{\tau'},\iota}[j] \neq \mathsf{cert}_{i^{(D)}_{\tau'},\iota}[j], \\ 0, & \text{otherwise.} \end{cases}$$

Then

$$\mu := \frac{1}{|\mathcal{J}^*_{i^{(D)}_{\tau'}}|} \sum_{j \in \mathcal{J}^*_{i^{(D)}_{\tau'}}} \zeta_j \geq \frac{m'_L}{2(m - \tau^*\ell)}.$$

Conditioned on this fixed $\mathbf{y}_{i^{(D)}_{\tau'},\iota}$, $\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}$ remains a uniformly random subset of $\mathcal{J}^*_{i^{(D)}_{\tau'}}$ of size $(n-\tau^*)\ell$. Therefore, by Hoeffding's inequality for sampling without replacement (Lemma 9),

$$\begin{aligned} &\Pr_{\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}} \left[\mathsf{cert}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}] = \mathbf{y}_{i^{(D)}_{\tau'},\iota}[\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}]\right] \\ &= \Pr\left[\sum_{j \in \mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}} \zeta_j = 0\right] \\ &\leq 2\exp\big(-2(n-\tau^*)\ell\mu^2\big) \leq 2\exp\left(-\frac{\ell {m'_L}^2}{2m^2}\right), \end{aligned}$$

where the last inequality uses $n - \tau^* \geq 1$ and $m - \tau^*\ell \leq m$. Since the bound holds for every $\mathbf{y}_{i^{(D)}_{\tau'},\iota}$ satisfying the Hamming-weight condition, averaging over $\mathbf{y}_{i^{(D)}_{\tau'},\iota}$ proves the lemma.

Remark 35. *The proof of Lemma 34 only uses the following sampling property. Fix an additional check with $\tau^* := |\mathcal{CS}| < n$ and the corresponding set $\mathcal{J}^*_{i^{(D)}_{\tau'}}$. Let $\{\zeta_j\}_{j \in \mathcal{J}^*_{i^{(D)}_{\tau'}}}$ be any possibly random binary variables such that, conditioned on the information determining these variables, $\mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}$ remains a uniformly random subset of $\mathcal{J}^*_{i^{(D)}_{\tau'}}$ of size $(n-\tau^*)\ell$. Then the same argument gives*

$$\Pr\left[\sum_{j \in \mathcal{J}^*_{i^{(D)}_{\tau'}}} \zeta_j \geq m'_L/2 \ \wedge\ \zeta_j = 0 \text{ for every } j \in \mathcal{J}^{*\times}_{i^{(D)}_{\tau'}}\right] \leq 2\exp\left(-\frac{\ell(m'_L)^2}{2m^2}\right).$$

We will use this form of the argument in Lemma 46, where ζ_j records disagreement in a fixed Fourier-basis coordinate after the MPC evaluation.

5 Construction of Outsourced MPC with Unanimously Verifiable Deletion for Arithmetic Circuits

In this section, we construct an outsourced MPC protocol with (t, n)-adaptively secure unanimously verifiable deletion for any arithmetic circuit, where t, n are positive integers such that $n \geq 2t + 1$.

In Section 5.1, we introduce some notation for arithmetic circuits. In Section 5.2, we present the construction of our outsourced MPC protocol. In Sections 5.3 and 5.4, we present the reconstruction correctness and security proof for the protocol of Section 5.2. Finally, Section 5.5 describes an efficiency improvement to the construction of Section 5.2.

5.1 Arithmetic Circuits

Every arithmetic circuit is composed of the following four kinds of gates:

- c-$\mathsf{sADD}(s)$: the scalar addition gate, characterized by a constant $c \in \mathbb{Z}_q$, takes in a value $s \in \mathbb{Z}_q$, and outputs $s + c$.
- $\mathsf{ADD}(s_1, s_2)$: the addition gate takes in two values s_1, s_2, and outputs $s_1 + s_2$.
- c-$\mathsf{sMUL}(s)$: the scalar multiplication gate, characterized by a constant $c \in \mathbb{Z}_q$, takes in a value $s \in \mathbb{Z}_q$, and outputs cs.
- $\mathsf{MUL}(s_1, s_2)$: the multiplication gate takes in two values s_1, s_2, and outputs $s_1 s_2$.

We refer to sADD, ADD, and sMUL as linear gates, and the circuits consisting only of linear gates as linear circuits. Without loss of generality, we assume that the two input wires of every binary gate are distinct. If the same wire feeds both inputs of a binary gate, we insert an identity gate (0-sADD) on one of the two incoming edges.

For any function $F : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n'}$, assume it can be computed by a circuit C with N gates. The gates are topologically sorted as $\mathsf{GATE}_1, \ldots, \mathsf{GATE}_N$, meaning the circuit can be evaluated by processing the gates in this order. We label the wires of the circuit as follows:

- For each $i \in [n]$, we label the i-th input wire by WIRE_i.
- For each $g \in [N]$, we label the output wire of GATE_g by WIRE_{g+n}.

Let N^{mul} be the number of multiplication gates. Let $\mathsf{GATE}_{l_1}, \ldots, \mathsf{GATE}_{l_{N^{\mathsf{mul}}}}$ be the multiplication gates in topological order, where $1 \le l_1 < l_2 < \cdots < l_{N^{\mathsf{mul}}} \le N$. To help illustrate these notations, we depict in Figure 1 the circuit corresponding to $F(x_1, x_2, x_3) := ((x_1 + c) + x_2) \cdot (c' x_3) \cdot x_2$ as an illustrative example.

Let n' be the number of outputs. Define $\Gamma := N^{\mathsf{mul}} + 1$ and $l_\Gamma := N + 1$. For each $\kappa \in [\Gamma]$, define

$$M_\kappa := \begin{cases} l_\kappa + n + 1 & \kappa \in [1, N^{\mathsf{mul}}], \\ n + N + n' & \kappa = \Gamma. \end{cases} \tag{18}$$

In our MPC protocol, Γ represents the number of secret-sharing batches, and M_κ represents the number of sharings in the κ-th batch.

5.2 Protocol Construction

In this section, we construct an outsourced MPC protocol for arbitrary arithmetic circuits. The core idea is to decompose the circuit into several subcircuits consisting only of linear gates. Figure 2

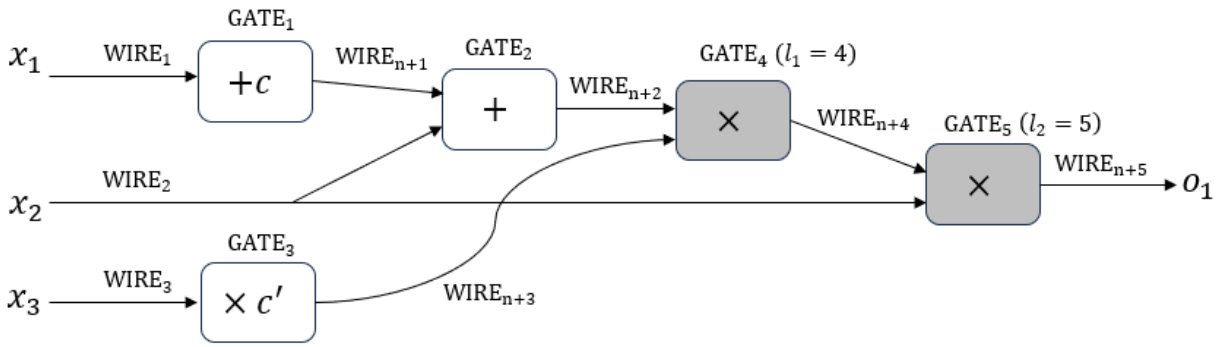


Figure 1: An arithmetic circuit computing $F(x_1, x_2, x_3) := ((x_1 + c) + x_2) \cdot (c'x_3) \cdot x_2$. In this example, the circuit has $n = 3$ inputs, $n' = 1$ output, $N = 5$ gates, and $N^{\mathsf{mul}} = 2$ multiplication gates.

illustrates this approach through the example $F(x_1, x_2, x_3) := ((x_1 + c) + x_2) \cdot (c'x_3) \cdot x_2$. Instead of evaluating the circuit in Figure 1 directly, one can compute F by evaluating, in sequence, the linear circuits in Figures 2a, 2b, and 2c. Moreover, each of these linear circuits is evaluated using a distinct secret-sharing batch.

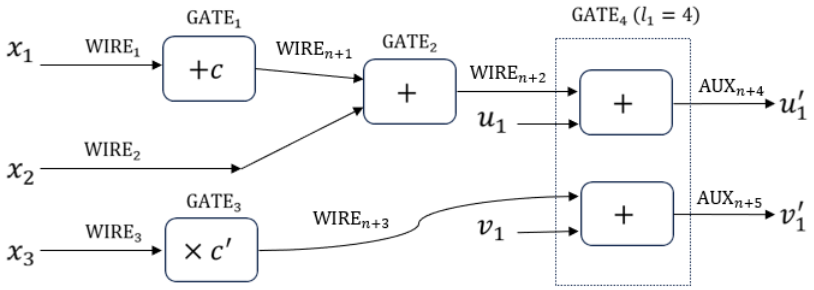
Below, we present the construction of an outsourced MPC protocol Π_F^{MPC} for a function $F : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n'}$ computed by an arbitrary polynomial-size arithmetic circuit. The protocol involves input clients $\mathcal{C}_1, \ldots, \mathcal{C}_n$ and computing parties (servers) $\mathcal{P}_1, \ldots, \mathcal{P}_n$. Each client $\mathcal{C}_i$ holds an input $x_i \in \mathbb{Z}_q$, while the computing parties hold no private inputs of their own. The clients participate only in the input phase and retire immediately after broadcasting their masked inputs. Throughout the remainder of this section, the term "parties" refers exclusively to the computing parties.

For any $s \in \mathbb{Z}_q$, any $\kappa \in [\Gamma]$, and any sharing position ι_κ, we use $[\![s]\!]_{\kappa,\iota_\kappa}$ to denote the quantum secret sharing with secret value s in the κ-th batch indexed by ι_κ. For every $a \in \mathbb{Z}_q$, we use $\langle a \rangle$ to denote a classical robust-openable n-out-of-n secret sharing. We use "$\Leftarrow$" to indicate a homomorphic update of a sharing, where the secret encoded by the sharing is changed according to the expression on the right-hand side.

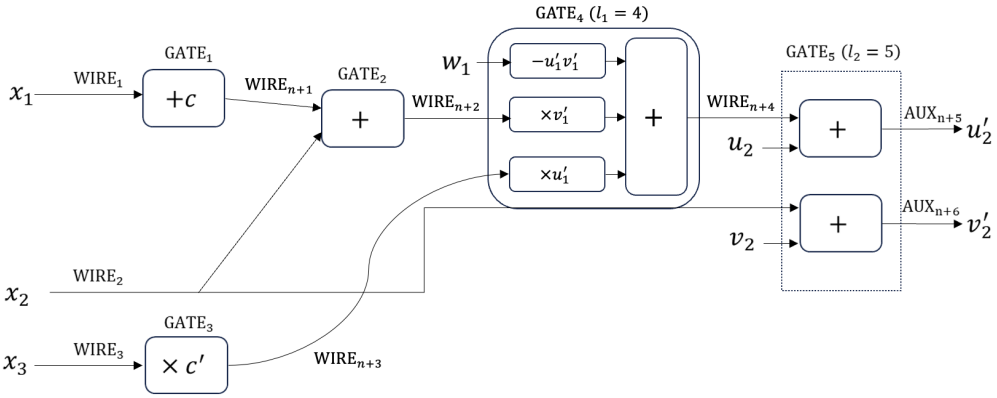
All online messages of both the input clients and the computing parties are sent through the classical synchronous authenticated broadcast channel specified in Remark 22. As in Section 4, let $q := 2^{\Theta(\lambda)}$ be the prime modulus, and adopt the secret-sharing parameters $m, \ell, m', m'_L, m'_R, d$ from Section 4.2. For every $\kappa \in [\Gamma]$, let $\Pi^{(\kappa)} := \Pi^{\mathsf{SSB-UVD}}_{n,t,M_\kappa,\mathbb{Z}_q}$ denote an $(n, t, M_\kappa, \mathbb{Z}_q)$-secret-sharing batch scheme defined in Section 4.3. Let Π_{ROpenSS} be a realization of the adaptively programmable classical secret-sharing protocol with robust opening defined in Definition 14. We insert remarks in italics led by "$\triangleright$" to describe the purpose of the main phases.

- **Offline phase.** In this phase, the trusted party $\mathcal{T}$ prepares the following:

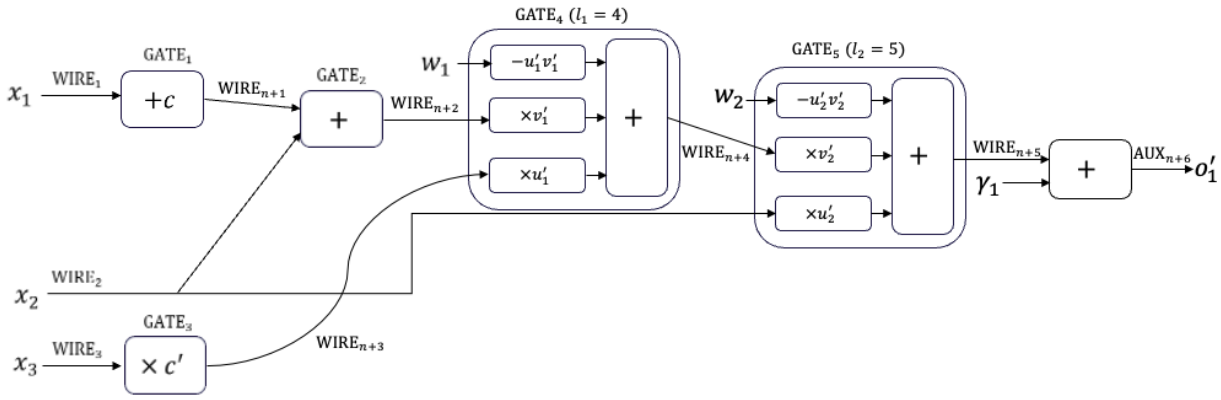
  1. **Setup for secret-sharing batches.** For every $\kappa \in [\Gamma]$, initialize the κ-th batch by calling
  $$\{\mathsf{mvk}^{(\kappa)}_{i_1,i_2}\}_{i_1,i_2\in[n]} \leftarrow \Pi^{(\kappa)}.\mathsf{Setup}(1^\lambda).$$
  2. **Preparation for the inputs.** For every $i^* \in [n]$, sample a uniformly random input mask $\Delta_{i^*} \leftarrow \mathbb{Z}_q$ and privately send Δ_{i^*} to the input client $\mathcal{C}_{i^*}$. For every $\kappa \in [\Gamma]$,



(a) $(u'_1, v'_1) := F^{(1)}(x_1, x_2, x_3, u_1, v_1) := (((x_1 + c) + x_2) + u_1, c'x_3 + v_1)$.



(b) $(u'_2, v'_2) := F^{(2)}(x_1, x_2, x_3, w_1, u_2, v_2; u'_1, v'_1) := ((w_1 - u'_1v'_1 + v'_1((x_1 + c) + x_2) + u'_1(c'x_3)) + u_2, x_2 + v_2)$.



(c) $o'_1 := F^{(3)}(x_1, x_2, x_3, w_1, w_2, \gamma_1; u'_1, v'_1, u'_2, v'_2) := w_2 - u'_2v'_2 + v'_2\big(w_1 - u'_1v'_1 + v'_1((x_1 + c) + x_2) + u'_1(c'x_3)\big) + u'_2x_2 + \gamma_1$.

Figure 2: Decomposition of the circuit in Figure 1 into $\Gamma = N^{\mathsf{mul}} + 1 = 3$ subcircuits, each computing a linear function. For each circuit, we construct a corresponding secret-sharing batch with $M_1 = n + l_1 + 1 = n + 5$, $M_2 = n + l_2 + 1 = n + 6$, and $M_3 = n + N + n' = n + 6$ sharings, respectively.

generate the i^*-th sharing in the κ-th batch by

$$\left(\{\mathsf{vk}^{(\kappa)}_{i_1,i_2,i^*}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}^{(\kappa)}_{i,i^*}\}_{i\in[n]}\right) \leftarrow \Pi^{(\kappa)}.\mathsf{Split}\left(i^*, \Delta_{i^*}, \{\mathsf{mvk}^{(\kappa)}_{i_1,i_2}\}_{i_1,i_2\in[n]}\right). \tag{19}$$

3. **Preparation for the gates.** For each $g \in [N]$, let $\iota := g + n$, and proceed as follows:
   - If GATE_g is a linear gate, let ν be the smallest integer such that $l_\nu > g$. For every $\kappa \in [\nu, \Gamma]$, generate the ι-th sharing in the κ-th batch by

   $$(\{\mathsf{vk}^{(\kappa)}_{i_1,i_2,\iota}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}^{(\kappa)}_{i,\iota}\}_{i\in[n]}) \leftarrow \Pi^{(\kappa)}.\mathsf{Split}(\iota, 0, \{\mathsf{mvk}^{(\kappa)}_{i_1,i_2}\}_{i_1,i_2\in[n]}).$$

   - If GATE_g is a multiplication gate, let ν be the integer such that $l_\nu = g$. Randomly sample $u_\nu, v_\nu \leftarrow \mathbb{Z}_q$ and let $w_\nu := u_\nu v_\nu$. Generate the ι-th and $(\iota+1)$-th sharings in the ν-th batch by

   $$(\{\mathsf{vk}^{(\nu)}_{i_1,i_2,\iota}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}^{(\nu)}_{i,\iota}\}_{i\in[n]}) \leftarrow \Pi^{(\nu)}.\mathsf{Split}(\iota, u_\nu, \{\mathsf{mvk}^{(\nu)}_{i_1,i_2}\}_{i_1,i_2\in[n]}),$$

   $$(\{\mathsf{vk}^{(\nu)}_{i_1,i_2,\iota+1}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}^{(\nu)}_{i,\iota+1}\}_{i\in[n]}) \leftarrow \Pi^{(\nu)}.\mathsf{Split}(\iota+1, v_\nu, \{\mathsf{mvk}^{(\nu)}_{i_1,i_2}\}_{i_1,i_2\in[n]}).$$

   For every $\kappa \in [\nu+1, \Gamma]$, generate the ι-th sharing in the κ-th batch by

   $$(\{\mathsf{vk}^{(\kappa)}_{i_1,i_2,\iota}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}^{(\kappa)}_{i,\iota}\}_{i\in[n]}) \leftarrow \Pi^{(\kappa)}.\mathsf{Split}(\iota, w_\nu, \{\mathsf{mvk}^{(\kappa)}_{i_1,i_2}\}_{i_1,i_2\in[n]}).$$

4. **Preparation for the outputs.** For every $h \in [n']$, sample a uniformly random output mask $\gamma_h \leftarrow \mathbb{Z}_q$ and let $\iota_h := n + N + h$. Generate the ι_h-th sharing in the Γ-th batch by

   $$\left(\{\mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota_h}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}^{(\Gamma)}_{i,\iota_h}\}_{i\in[n]}\right) \leftarrow \Pi^{(\Gamma)}.\mathsf{Split}\left(\iota_h, \gamma_h, \{\mathsf{mvk}^{(\Gamma)}_{i_1,i_2}\}_{i_1,i_2\in[n]}\right). \tag{20}$$

   In addition, $\mathcal{T}$ invokes

   $$\langle\gamma_h\rangle \leftarrow \Pi_{\mathsf{ROpenSS}}.\mathsf{Split}(1^\lambda, \gamma_h). \tag{21}$$

5. **Packing.** For every $i_1 \in [n]$, let

   $$\mathsf{pdata}_{i_1} := \left\{\mathsf{sh}^{(\kappa)}_{i_1,\iota_\kappa}\right\}_{\kappa\in[\Gamma],\,\iota_\kappa\in[M_\kappa]},$$

   and let

   $$\mathsf{ver_key}_{i_1} := \left(\{\mathsf{mvk}^{(\kappa)}_{i_1,i_2}\}_{i_2\in[n],\,\kappa\in[\Gamma]}, \{\mathsf{vk}^{(\kappa)}_{i_1,i_2,\iota_\kappa}\}_{i_2\in[n],\,\kappa\in[\Gamma],\,\iota_\kappa\in[M_\kappa]}\right).$$

   Give $(\mathsf{pdata}_{i_1}, \mathsf{ver_key}_{i_1})$ to $\mathcal{P}_{i_1}$. The local shares generated by $\{\langle\gamma_h\rangle\}_{h\in[n']}$ are distributed among $\mathcal{P}_1, \ldots, \mathcal{P}_n$ according to Π_{ROpenSS}.
6. Initialize the adaptive deletion set $\mathcal{DS} \leftarrow \emptyset$.
7. Through out the protocol, we parse $\mathsf{sh}^{(\kappa)}_{i_1,\iota} = \left(|\psi^{(\kappa)}_{i_1,\iota}\rangle, \sigma^{(\kappa)}_\iota\right)$ for every $i_1 \in [n],\ \kappa \in [\Gamma],\ \iota \in [M_\kappa]$.[4]

[4] All classical masks $\sigma^{(\kappa)}_\iota$ are public values, whose initial values are published by $\mathcal{T}$ during preprocessing. Every party wishing to obtain the correct output, including a party whose quantum shares have been deleted, must perform the prescribed classical updates to these masks. Accordingly, the conditions $i_1 \notin \mathcal{DS}$ below restrict only quantum operations, not the classical mask updates.

▷ *For every* $\kappa \in [\Gamma], \iota_\kappa \in [M_\kappa]$*, define*

$$\textit{If } \kappa \in [N^{\mathsf{mul}}],\ \mathsf{val}_{\kappa,\iota_\kappa} := \begin{cases} \Delta_{\iota_\kappa}, & \iota_\kappa \in [n], \\ 0, & \iota_\kappa = n+g,\ g \in [l_\kappa - 1],\ \textit{and } \mathsf{GATE}_g \textit{ is linear}, \\ w_\nu, & \iota_\kappa = n + l_\nu,\ \nu \in [\kappa - 1], \\ u_\kappa, & \iota_\kappa = M_\kappa - 1, \\ v_\kappa, & \iota_\kappa = M_\kappa. \end{cases} \tag{22}$$

$$\textit{If } \kappa = \Gamma,\ \mathsf{val}_{\kappa,\iota_\kappa} := \begin{cases} \Delta_{\iota_\kappa}, & \iota_\kappa \in [n], \\ 0, & \iota_\kappa = n+g,\ g \in [N],\ \textit{and } \mathsf{GATE}_g \textit{ is linear}, \\ w_\nu, & \iota_\kappa = n + l_\nu,\ \nu \in [N^{\mathsf{mul}}], \\ \gamma_h, & \iota_\kappa = n + N + h,\ h \in [n']. \end{cases}$$

In the offline phase, $\mathcal{T}$ *distributes*

$$\{[\![\mathsf{val}_{\kappa,\iota_\kappa}]\!]_{\kappa,\iota_\kappa}\}_{\kappa\in[\Gamma],\iota_\kappa\in[M_\kappa]}, \{\langle\gamma_h\rangle\}_{h\in[n']}$$

among $\mathcal{P}_1, \ldots, \mathcal{P}_n$*, and privately sends the input mask* Δ_i *to client* $\mathcal{C}_i$ *for every* $i \in [n]$*.*

- **Input phase.** The input clients and computing parties proceed as follows:

  1. For every $i^* \in [n]$, the input client $\mathcal{C}_{i^*}$ broadcasts

  $$\Delta'_{i^*} := x_{i^*} - \Delta_{i^*}. \tag{23}$$

  If a required client broadcast is missing, every honest computing party aborts. After broadcasting its input message, $\mathcal{C}_{i^*}$ retires and takes no further part in the protocol.

  2. For every $i_1 \in [n]$ and every $i^* \in [n]$, upon receiving Δ'_{i^*}, if $i_1 \notin \mathcal{DS}$, computing party $\mathcal{P}_{i_1}$ computes, for every $\kappa \in [\Gamma]$,

  $$\widetilde{\mathsf{sh}}^{(\kappa)}_{i_1,i^*} := \Pi^{(\kappa)}.\mathsf{AddSh}\left(\Delta'_{i^*}, i_1, i^*, \mathsf{sh}^{(\kappa)}_{i_1,i^*}\right) \tag{24}$$

  and updates $\mathsf{sh}^{(\kappa)}_{i_1,i^*} \leftarrow \widetilde{\mathsf{sh}}^{(\kappa)}_{i_1,i^*}$.

▷ *In the input phase, for every* $\kappa \in [\Gamma]$ *and* $i^* \in [n]$*, the parties compute*

$$[\![x_{i^*}]\!]_{\kappa,i^*} \Leftarrow [\![\Delta_{i^*}]\!]_{\kappa,i^*} + \Delta'_{i^*}.$$

- **Evaluation phase.** At this point, all input clients have retired, and the remaining protocol is executed exclusively by the computing parties. The circuit evaluation proceeds by Γ rounds, which are referred to by Round_1, Round_2, ..., Round_Γ. In $\mathsf{Round}_{\mathsf{cur}}$ for $\mathsf{cur} \in [\Gamma]$ (cur abbreviates "current"), the parties enumerate $g = 1, 2, \cdots, \min\{N, l_{\mathsf{cur}}\}$, and evaluate GATE_g as follows:

  – **Scalar addition:** If GATE_g is a c-sADD gate. Let the input wire be WIRE_{ι_1}. Let $\iota_0 := g + n$, and then WIRE_{ι_0} is the output wire of GATE_g. For every $i_1 \in [n]$, $\mathcal{P}_{i_1}$ does the following:

1. For each $i_2 \in [n]$, $\mathcal{P}_{i_1}$ updates $\mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}$ to

$$(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}) \\ := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}(1, i_1, i_2, \iota_1, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})$$

2. If $i_1 \notin \mathcal{DS}$, $\mathcal{P}_{i_1}$ computes

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_1}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}(1, i_1, \iota_1, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_1}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_0}),$$

$$\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0} := \Pi^{(\mathsf{cur})}.\mathsf{AddSh}(c, i_1, \iota_0, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}).$$

Then, it updates $\mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_b}$ to $\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_b}$ for each $b \in \{0,1\}$.

▷ *When* $\mathsf{GATE}_g = c$-sADD*, the parties compute*

$$[\![s + c]\!]_{\mathsf{cur},\iota_0} \Leftarrow [\![0]\!]_{\mathsf{cur},\iota_0} + [\![s]\!]_{\mathsf{cur},\iota_1} + c.$$

– **Addition:** If GATE_g is an ADD gate, let the input wires be $\mathsf{WIRE}_{\iota_1}, \mathsf{WIRE}_{\iota_2}$. Let $\iota_0 := g + n$, and then WIRE_{ι_0} is the output wire of GATE_g. For every $i_1 \in [n]$, $\mathcal{P}_{i_1}$ does the following:

1. For each $i_2 \in [n]$, $\mathcal{P}_{i_1}$ computes

$$(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \overline{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}) \\ := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}(1, i_1, i_2, \iota_1, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})$$

$$(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_2}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}) \\ := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}(1, i_1, i_2, \iota_2, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_2}, \overline{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})$$

Then, it updates $\mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_b}$ to $\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_b}$ for each $b \in \{0,1,2\}$.

2. If $i_1 \notin \mathcal{DS}$, $\mathcal{P}_{i_1}$ computes

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_1}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}(1, i_1, \iota_1, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_1}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_0}),$$

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_2}, \widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}(1, i_1, \iota_2, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_2}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}),$$

Then, it updates $\mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_b}$ to $\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_b}$ for each $b \in \{0,1,2\}$.

▷ *When* $\mathsf{GATE}_g = \mathsf{ADD}$*, the parties compute*

$$[\![s_1 + s_2]\!]_{\mathsf{cur},\iota_0} \Leftarrow [\![0]\!]_{\mathsf{cur},\iota_0} + [\![s_1]\!]_{\mathsf{cur},\iota_1} + [\![s_2]\!]_{\mathsf{cur},\iota_2}.$$

– **Scalar multiplication:** If GATE_g is a c-sMUL gate, let the input wire be WIRE_{ι_1}. Let $\iota_0 := g + n$, and then WIRE_{ι_0} is the output wire of GATE_g. For every $i_1 \in [n]$, $\mathcal{P}_{i_1}$ does the following:

1. For each $i_2 \in [n]$, $\mathcal{P}_{i_1}$ updates $\mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}$ to

$$(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}) \\ := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}(c, i_1, i_2, \iota_1, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})$$

2. If $i_1 \notin \mathcal{DS}$, $\mathcal{P}_{i_1}$ updates $\mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_1}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_0}$ to

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_1}, \widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}(c, i_1, \iota_1, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_1}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_0}),$$

▷ *When* $\mathsf{GATE}_g = c$-sMUL*, the parties compute*

$$[\![cs]\!]_{\mathsf{cur},\iota_0} \Leftarrow [\![0]\!]_{\mathsf{cur},\iota_0} + c \cdot [\![s]\!]_{\mathsf{cur},\iota_1}.$$

– **Multiplication:** If GATE_g is a MUL gate,

- If $g = l_{\mathsf{cur}}$, let the input wires be $\mathsf{WIRE}_{\iota_1}, \mathsf{WIRE}_{\iota_2}$. Let $\iota_3 = M_{\mathsf{cur}} - 1, \iota_4 = M_{\mathsf{cur}}$, and denote by $\mathsf{AUX}_{\iota_3}, \mathsf{AUX}_{\iota_4}$ the two auxiliary wires containing random masks of the cur-th beaver triple in the cur-th linear circuit. For every $i_1 \in [n]$, $\mathcal{P}_{i_1}$ does the following:
  1. For each $b \in \{1, 2\}$, each $i_2 \in [n]$, $\mathcal{P}_{i_1}$ updates $\mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_b}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_{b+2}}$ to

$$(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_b}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_{b+2}}) \\ := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}(1, i_1, i_2, \iota_b, \iota_{b+2}, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_b}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_{b+2}})$$

  2. If $i_1 \notin \mathcal{DS}$, for each $b \in \{1, 2\}$, $\mathcal{P}_{i_1}$ updates $\mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_b}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_{b+2}}$ to

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_b}, \widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_{b+2}}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}(1, i_1, \iota_b, \iota_{b+2}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_b}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_{b+2}}).$$

  3. If $i_1 \notin \mathcal{DS}$, for every $\iota \in [1, M_{\mathsf{cur}} - 2]$, run $\mathsf{cert}^{(\mathsf{cur})}_{i_1,\iota} \leftarrow \Pi^{(\mathsf{cur})}.\mathsf{Delete}(i_1, \iota, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota})$. Broadcast $\mathsf{rec_cert}^{(\mathsf{cur})}_{i_1} := \{\mathsf{cert}^{(\mathsf{cur})}_{i_1,\iota}\}_{\iota \in [1, M_{\mathsf{cur}}-2]}$.
  4. $\mathcal{P}_{i_1}$, after receiving $\mathsf{rec_cert}^{(\mathsf{cur})}_{i_2} = \{\mathsf{cert}^{(\mathsf{cur})}_{i_2,\iota}\}_{\iota \in [1, M_{\mathsf{cur}}-2]}$ from each $\mathcal{P}_{i_2}$ such that $i_2 \notin \mathcal{DS}$, checks whether

$$\Pi^{(\mathsf{cur})}.\mathsf{Verify}(i_1, i_2, \iota, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota}, \mathsf{cert}^{(\mathsf{cur})}_{i_2,\iota}) = \top$$

  for each $\iota \in [1, M_{\mathsf{cur}} - 2]$. If each $\mathcal{P}_{i_2}$ passes this check, then $\mathcal{P}_{i_1}$ broadcasts $\top$. Otherwise, $\mathcal{P}_{i_1}$ broadcasts $\bot$.
  5. $\mathcal{P}_{i_1}$ aborts immediately if it receives $\bot$ from any party. Otherwise, after it receives $\top$ from $\mathcal{P}_{i_2}$ for every $i_2 \in [n]$, if $i_1 \notin \mathcal{DS}$, $\mathcal{P}_{i_1}$ measures $|\boldsymbol{\psi}^{(\mathsf{cur})}_{i_1,\iota_3}\rangle$ and $|\boldsymbol{\psi}^{(\mathsf{cur})}_{i_1,\iota_4}\rangle$ on the computational basis and get the results $\mathbf{z}^{(\mathsf{cur})}_{i_1,\iota_3}, \mathbf{z}^{(\mathsf{cur})}_{i_1,\iota_4} \in \mathbb{Z}_q^m$. Then it broadcasts $(\mathbf{z}^{(\mathsf{cur})}_{i_1,\iota_3}, \mathbf{z}^{(\mathsf{cur})}_{i_1,\iota_4})$.
  6. $\mathcal{P}_{i_1}$, after receiving $\mathbf{z}^{(\mathsf{cur})}_{i_2,\iota_3}, \mathbf{z}^{(\mathsf{cur})}_{i_2,\iota_4}$ from each $\mathcal{P}_{i_2}$ such that $i_2 \notin \mathcal{DS}$, applies polynomial interpolation with error correction (Lemma 8) to $\{\mathbf{z}^{(\mathsf{cur})}_{i_2,\iota_3}\}_{i_2 \in [n] \setminus \mathcal{DS}}$ and

$\{\mathbf{z}^{(\mathsf{cur})}_{i_2,\iota_4}\}_{i_2\in[n]\setminus\mathcal{DS}}$. Let $\widehat{f}^{(\mathsf{cur})}_{\iota_3}$ and $\widehat{f}^{(\mathsf{cur})}_{\iota_4}$ denote the degree-at-most-d polynomials returned by the two decoders. If either decoder outputs $\perp$, it aborts. Otherwise, for every $i_2 \in [n]\setminus\mathcal{DS}$ and $b \in \{3,4\}$, it checks whether

$$\left|\left\{j \in [m] : \mathbf{z}^{(\mathsf{cur})}_{i_2,\iota_b}[j] \neq \widehat{f}^{(\mathsf{cur})}_{\iota_b}((i_2-1)m+j)\right\}\right| \leq n\ell.$$

If any check fails, it aborts. Otherwise, it reconstructs u'_{cur} from $\widehat{f}^{(\mathsf{cur})}_{\iota_3}$ and $\sigma^{(\mathsf{cur})}_{\iota_3}$, and reconstructs v'_{cur} from $\widehat{f}^{(\mathsf{cur})}_{\iota_4}$ and $\sigma^{(\mathsf{cur})}_{\iota_4}$.

▷ *If $g = l_{\mathsf{cur}}$, then it is the last* MUL *gate in the* cur*-th batch and* $\mathsf{cur} \leq N^{\mathsf{mul}}$*. The parties compute*

$$[\![u'_{\mathsf{cur}}]\!]_{\mathsf{cur},\iota_3} \Leftarrow [\![u_{\mathsf{cur}}]\!]_{\mathsf{cur},\iota_3} + [\![s_1]\!]_{\mathsf{cur},\iota_1},$$

and

$$[\![v'_{\mathsf{cur}}]\!]_{\mathsf{cur},\iota_4} \Leftarrow [\![v_{\mathsf{cur}}]\!]_{\mathsf{cur},\iota_4} + [\![s_2]\!]_{\mathsf{cur},\iota_2}.$$

Then, they delete all the shares in the cur*-th batch except $[\![u'_{\mathsf{cur}}]\!]_{\mathsf{cur},\iota_3}, [\![v'_{\mathsf{cur}}]\!]_{\mathsf{cur},\iota_4}$, and reconstruct $u'_{\mathsf{cur}}, v'_{\mathsf{cur}}$.*

- If $g \neq l_{\mathsf{cur}}$, there exists an integer $\nu \in [1, \mathsf{cur}-1]$ such that $g = l_\nu$. Let u'_ν, v'_ν be the values reconstructed at Round_ν. Let the input wires be $\mathsf{WIRE}_{\iota_1}, \mathsf{WIRE}_{\iota_2}$. Let $\iota_0 := g + n$, and then WIRE_{ι_0} is the output wire of GATE_g. For every $i_1 \in [n]$, $\mathcal{P}_{i_1}$ does the following:
  1. For each $i_2 \in [n]$, $\mathcal{P}_{i_1}$ computes

$$\begin{aligned}&(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \overline{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})\\ &:= \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}(v'_\nu, i_1, i_2, \iota_1, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})\end{aligned}$$

$$\begin{aligned}&(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_2}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})\\ &:= \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}(u'_\nu, i_1, i_2, \iota_2, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_2}, \overline{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})\end{aligned}$$

     Then, it updates $\mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_b}$ to $\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_b}$ for each $b \in \{0,1,2\}$.
  2. If $i_1 \notin \mathcal{DS}$, $\mathcal{P}_{i_1}$ computes

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_1}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}(v'_\nu, i_1, \iota_1, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_1}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_0}),$$

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_2}, \widehat{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}(u'_\nu, i_1, \iota_2, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_2}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}),$$

$$\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0} := \Pi^{(\mathsf{cur})}.\mathsf{AddSh}(-u'_\nu v'_\nu, i_1, \iota_0, \widehat{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}).$$

     Then, for each $b \in \{0,1,2\}$, it updates $\mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_b}$ to $\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_b}$.

▷ *If* $\mathsf{GATE}_g = \mathsf{MUL}$ *but* $g \neq l_{\mathsf{cur}}$*, the parties compute*

$$[\![s_1 s_2]\!]_{\mathsf{cur},\iota_0} \Leftarrow [\![w_\nu]\!]_{\mathsf{cur},\iota_0} + v'_\nu \cdot [\![s_1]\!]_{\mathsf{cur},\iota_1} + u'_\nu \cdot [\![s_2]\!]_{\mathsf{cur},\iota_2} - u'_\nu v'_\nu.$$

– In Round_Γ, after evaluating all the gates, the parties still need to compute (the sharing of) the addition of the output and their masks. Concretely, assume the output wires are $\mathsf{WIRE}_{\iota_1^{(o)}}, \ldots, \mathsf{WIRE}_{\iota_{n'}^{(o)}}$. For every $h \in [n']$, let $\iota_h = n + N + h$. For every $i_1 \in [n]$, $\mathcal{P}_{i_1}$ does the following:

  * For each $i_2 \in [n]$, every $h \in [n']$, $\mathcal{P}_{i_1}$ updates $\mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota_h^{(o)}}, \mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota_h}$ to

$$\begin{aligned}&(\widetilde{\mathsf{vk}}^{(\Gamma)}_{i_1,i_2,\iota_h^{(o)}}, \widetilde{\mathsf{vk}}^{(\Gamma)}_{i_1,i_2,\iota_h})\\ &:= \Pi^{(\Gamma)}.\mathsf{ScaleAddVk}(1, i_1, i_2, \iota_h^{(o)}, \iota_h, \mathsf{mvk}^{(\Gamma)}_{i_1,i_2}, \mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota_h^{(o)}}, \mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota_h})\end{aligned}$$

  * If $i_1 \notin \mathcal{DS}$, for every $h \in [n']$, $\mathcal{P}_{i_1}$ updates $\mathsf{sh}^{(\Gamma)}_{i_1,\iota_h^{(o)}}, \mathsf{sh}^{(\Gamma)}_{i_1,\iota_h}$ to

$$(\widetilde{\mathsf{sh}}^{(\Gamma)}_{i_1,\iota_h^{(o)}}, \widetilde{\mathsf{sh}}^{(\Gamma)}_{i_1,\iota_h}) := \Pi^{(\Gamma)}.\mathsf{ScaleAddSh}(1, i_1, \iota_h^{(o)}, \iota_h, \mathsf{sh}^{(\Gamma)}_{i_1,\iota_h^{(o)}}, \mathsf{sh}^{(\Gamma)}_{i_1,\iota_h}).$$

  ▷ *For every $h \in [n']$, let o_h denote the h-th output. At the end of Round_Γ, the parties compute*

$$[\![o_h + \gamma_h]\!]_{\Gamma,\iota_h} \Leftarrow [\![\gamma_h]\!]_{\Gamma,\iota_h} + [\![o_h]\!]_{\Gamma,\iota_h^{(o)}}. \tag{25}$$

- **Adaptive deletion for $\mathcal{P}_{i_1}$.** At an adaptive-query point before $\mathsf{Round}_{\mathsf{cur}}$ (we regard the initial adaptive-query point before the input phase as the adaptive-query point before Round_1), when $\mathcal{P}_{i_1}$ is instructed to delete its remaining quantum state, the parties proceed as follows:

  1. For every $\kappa \in [\mathsf{cur}, \Gamma]$ and $\iota_\kappa \in [M_\kappa]$, $\mathcal{P}_{i_1}$ runs

$$\mathsf{cert}^{(\kappa)}_{i_1,\iota_\kappa} \leftarrow \Pi^{(\kappa)}.\mathsf{Delete}(i_1, \iota_\kappa, \mathsf{sh}^{(\kappa)}_{i_1,\iota_\kappa})$$

     and broadcasts

$$\mathsf{del_cert}_{i_1} := \{\mathsf{cert}^{(\kappa)}_{i_1,\iota_\kappa}\}_{\kappa\in[\mathsf{cur},\Gamma],\iota_\kappa\in[M_\kappa]}.$$

  2. Every $\mathcal{P}_{i_2}$ verifies all certificates using the corresponding verification keys. It broadcasts $\top$ if all verifications accept and broadcasts $\bot$ otherwise. Every party aborts if it receives $\bot$ from any party.
  3. Set $\mathcal{DS} \leftarrow \mathcal{DS} \cup \{i_1\}$.

▷ *The adaptive deletion procedure erases the quantum secret-sharing batches held by $\mathcal{P}_{i_1}$.*

- **Output phase.** For $h \in [n']$, let $\iota_h = n + N + h$. For every $i_1 \in [n]$, $\mathcal{P}_{i_1}$ does the following:

  1. If $i_1 \notin \mathcal{DS}$, run $\mathsf{cert}^{(\Gamma)}_{i_1,\iota} \leftarrow \Pi^{(\Gamma)}.\mathsf{Delete}(i_1, \iota, \mathsf{sh}^{(\Gamma)}_{i_1,\iota})$ for every $\iota \in [n+N]$. Broadcast $\mathsf{rec_cert}^{(\Gamma)}_{i_1} := \{\mathsf{cert}^{(\Gamma)}_{i_1,\iota}\}_{\iota\in[n+N]}$.
  2. $\mathcal{P}_{i_1}$, after receiving $\mathsf{rec_cert}^{(\Gamma)}_{i_2} = \{\mathsf{cert}^{(\Gamma)}_{i_2,\iota}\}_{\iota\in[n+N]}$ from each $\mathcal{P}_{i_2}$ such that $i_2 \notin \mathcal{DS}$, checks whether

$$\Pi^{(\Gamma)}.\mathsf{Verify}(i_1, i_2, \iota, \mathsf{mvk}^{(\Gamma)}_{i_1,i_2}, \mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota}, \mathsf{cert}^{(\Gamma)}_{i_2,\iota}) = \top$$

     for every $\iota \in [n+N]$. If each $\mathcal{P}_{i_2}$ passes this check, then $\mathcal{P}_{i_1}$ broadcasts $\top$. Otherwise, $\mathcal{P}_{i_1}$ broadcasts $\bot$.

3. $\mathcal{P}_{i_1}$ aborts immediately if it receives $\perp$ from any party. Otherwise, after it receives $\top$ from every $\mathcal{P}_{i_2}$, if $i_1 \notin \mathcal{DS}$, it measures $|\psi^{(\Gamma)}_{i_1,\iota_h}\rangle$ in the computational basis and obtains $\mathbf{z}^{(\Gamma)}_{i_1,\iota_h} \in \mathbb{Z}_q^m$ for every $h \in [n']$. Then it broadcasts $\{\mathbf{z}^{(\Gamma)}_{i_1,\iota_h}\}_{h\in[n']}$.

4. $\mathcal{P}_{i_1}$, after receiving $\{\mathbf{z}^{(\Gamma)}_{i_2,\iota_h}\}_{h\in[n']}$ from each $\mathcal{P}_{i_2}$ such that $i_2 \notin \mathcal{DS}$, applies polynomial interpolation with error correction (Lemma 8) to $\{\mathbf{z}^{(\Gamma)}_{i_2,\iota_h}\}_{i_2\in[n]\setminus\mathcal{DS}}$ for every $h \in [n']$. Let $\widehat{f}^{(\Gamma)}_{\iota_h}$ denote the degree-at-most-d polynomial returned by the decoder for the h-th reconstruction. If any decoder outputs $\perp$, it aborts. Otherwise, for every $h \in [n']$ and $i_2 \in [n]\setminus\mathcal{DS}$, it checks whether

$$\left|\left\{j \in [m] : \mathbf{z}^{(\Gamma)}_{i_2,\iota_h}[j] \neq \widehat{f}^{(\Gamma)}_{\iota_h}((i_2-1)m+j)\right\}\right| \leq n\ell.$$

   If any check fails, it aborts. Otherwise, it reconstructs o'_h from $\widehat{f}^{(\Gamma)}_{\iota_h}$ and $\sigma^{(\Gamma)}_{\iota_h}$ for every $h \in [n']$.

5. For every $h \in [n']$, the parties invoke

$$\gamma_h \leftarrow \Pi_{\mathsf{ROpenSS}}.\mathsf{Open}(\langle\gamma_h\rangle).$$

   They abort if the opening returns $\perp$; otherwise, they compute

$$o_h := o'_h - \gamma_h. \tag{26}$$

6. If no abort occurs, every computing party outputs $(o_1, \ldots, o_{n'})$, and the protocol is declared successfully finalized. Since the output is publicly reconstructed through the broadcast channel, it is also available to the input clients.

▷ *In the output phase, the parties delete all sharings in the Γ-th batch except $\{[\![o'_h]\!]_{\Gamma,\iota_h}\}_{h\in[n']}$ and reconstruct $\{o'_h\}_{h\in[n']}$. They then robustly open $\langle\gamma_h\rangle$ and recover $o_h = o'_h - \gamma_h$ for every $h \in [n']$.*

In our outsourced MPC construction, we prepare a quantum sharing of 0 the output wire of every linear gate of the circuit. These zero sharings can in fact be eliminated to reduce the space complexity of the protocol. We include them here just to align the protocol more closely with the circuit structure, thereby making the subsequent simulator-based security proof more direct and easier to understand.

5.3 Reconstruction Correctness

We next prove that a malicious computing party cannot cause the honest computing parties to reconstruct an incorrect value. Since the external input clients are semi-honest and follow the prescribed input procedure, correctness is defined with respect to the input vector $\mathbf{x} := (x_1, \ldots, x_n)$ submitted by the clients.

Lemma 36 (Safe Reconstruction against Malicious Computing Parties)**.** *Consider any execution of Π_F^{MPC} that reaches a reconstruction step. Let $\mathcal{CS}$ and $\mathcal{DS}$ denote, respectively, the current sets of corrupted and deleted computing parties. Suppose that*

$$|[n]\setminus\mathcal{DS}| - |\mathcal{CS}\setminus\mathcal{DS}| \geq t+1.$$

Then every reconstruction performed by the honest computing parties either returns the correct value or causes the protocol to abort. More concretely:

- *for every $\nu \in [N^{\mathsf{mul}}]$, the reconstruction in Round_ν either returns the correct values u'_ν, v'_ν encoded by $[\![u'_\nu]\!]_{\nu,M_\nu-1}$ and $[\![v'_\nu]\!]_{\nu,M_\nu}$, or causes the protocol to abort; and*
- *for every $h \in [n']$, the reconstruction in the output phase either returns the correct masked output $o'_h = o_h + \gamma_h$ encoded by $[\![o'_h]\!]_{\Gamma,\iota_h}$, or causes the protocol to abort, where $(o_1, \ldots, o_{n'}) = F(x_1, \ldots, x_n)$.*

Moreover, whenever reconstruction succeeds, all honest computing parties obtain the same reconstructed value.

Proof. Conditioned on no previous abort, we first observe that, before every reconstruction, the corresponding registers of the honest undeleted computing parties form valid shares of the intended value under a common polynomial of degree at most d. This follows by induction over the protocol execution. The claim initially follows from the correctness of the preprocessing phase. In the input phase, adding the prescribed public value $\Delta'_i = x_i - \Delta_i$ transforms every input-mask sharing $[\![\Delta_i]\!]$ into a sharing $[\![x_i]\!]$ of the client's actual input. The claim is then preserved by the correctness of AddSh and $\mathsf{ScaleAddSh}$ for every linear gate. Finally, whenever a multiplication is evaluated, the inductive hypothesis and the correctness of all previous non-aborting reconstructions ensure that the public values u'_ν, v'_ν used in subsequent rounds are the values encoded by the corresponding sharings. Thus, deviations by corrupted computing parties can affect their own broadcast reconstruction strings, but cannot change the polynomial encoded by the honest computing parties' registers.

Fix any reconstruction, and let f be the degree-at-most-d polynomial underlying the sharing being reconstructed. If the decoder outputs $\perp$, the protocol aborts. Otherwise, let $\widehat{f}$ be the degree-at-most-d polynomial returned by the decoder. The protocol then performs the consistency checks specified in the reconstruction procedure. If any check fails, the protocol aborts.

Suppose that all consistency checks pass. Let

$$\mathcal{H} := ([n] \backslash \mathcal{DS}) \backslash (\mathcal{CS} \backslash \mathcal{DS})$$

denote the set of honest undeleted computing parties. By assumption, $|\mathcal{H}| \geq t+1$. For every $i \in \mathcal{H}$, let $\mathbf{z}_i \in \mathbb{Z}_q^m$ be its broadcast reconstruction string. Since $\mathcal{P}_i$ is honest, $\mathbf{z}_i$ agrees with the evaluations of f at every evaluation position of its share, and there are only $n\ell$ checking positions. Hence,

$$|\{j \in [m] : \mathbf{z}_i[j] \neq f((i-1)m + j)\}| \leq n\ell.$$

On the other hand, since the consistency check for $\mathcal{P}_i$ passes,

$$\left|\left\{j \in [m] : \mathbf{z}_i[j] \neq \widehat{f}((i-1)m + j)\right\}\right| \leq n\ell.$$

Therefore, f and $\widehat{f}$ agree on at least $m - 2n\ell$ points corresponding to $\mathcal{P}_i$. Since the evaluation points corresponding to different parties are distinct, $f - \widehat{f}$ has at least

$$|\mathcal{H}|(m - 2n\ell) \geq (t+1)(m - 2n\ell)$$

distinct roots.

In the proof of Theorem 24, we established

$$d < (t+1)(m-2n\ell).$$

Since both f and $\widehat{f}$ have degree at most d, the polynomial $f - \widehat{f}$ has degree at most d but more than d distinct roots. It follows that $f = \widehat{f}$. Hence, whenever reconstruction does not abort, the reconstructed value is the value encoded by the honest parties' shares.

5.4 Security Proof

In this section we present the security proof.

Theorem 37. *Suppose that Π_{ROpenSS} is an adaptively programmable n-out-of-n classical secret-sharing protocol with robust opening. Then Π_F^{MPC} securely realizes F in the outsourced-MPC model of Section 3.2, with adaptive semi-honest corruption of at most $n-1$ clients, (t,n)-adaptively secure unanimously verifiable deletion, and post-execution full corruption of the computing parties.*

Proof. Without loss of generality, we assume that $\mathcal{A}$ eventually corrupts every computing party after successful finalization. Any adversary making fewer such queries can be extended to collect the remaining residual states before terminating, and its original view is recovered by discarding these additional disclosures. To prove the theorem, we define a sequence of hybrid experiments $\mathsf{Hyb}_0(F,\mathbf{x})$, $\mathsf{Hyb}_1(F,\mathbf{x})$, $\mathsf{Hyb}_2(F,\mathbf{x})$, and $\mathsf{Sim}_2(F,\mathbf{x})$. In particular, $\mathsf{Hyb}_0(F,\mathbf{x})$ is the real-world execution of Π_F^{MPC}, while $\mathsf{Sim}_2(F,\mathbf{x})$ is an ideal-world execution generated by a simulator that is not initially given the clients' inputs and learns x_i whenever client $\mathcal{C}_i$ is adaptively corrupted. By Lemmas 38, 39, and 40, and by the triangle inequality,

$$\begin{aligned}
&\mathsf{TD}\big(\mathsf{Hyb}_0(F,\mathbf{x}), \mathsf{Sim}_2(F,\mathbf{x})\big) \\
&\le \mathsf{TD}\big(\mathsf{Hyb}_0(F,\mathbf{x}), \mathsf{Hyb}_1(F,\mathbf{x})\big) + \mathsf{TD}\big(\mathsf{Hyb}_1(F,\mathbf{x}), \mathsf{Hyb}_2(F,\mathbf{x})\big) + \mathsf{TD}\big(\mathsf{Hyb}_2(F,\mathbf{x}), \mathsf{Sim}_2(F,\mathbf{x})\big) \\
&= \mathrm{negl}(\lambda).
\end{aligned}$$

□

We next define the hybrid experiments and prove the three lemmas used above. At a high level, Hyb_1 changes the sampling order of the clients' input masks and public input messages, Hyb_2 replaces the protocol output by the output obtained from the ideal functionality using the programmable-completion property of Π_{ROpenSS}, and Sim_2 further replaces the secret values underlying all quantum secret-sharing batches by independent uniform values. To prove the last transition, we introduce auxiliary pairs of experiments starting from $\mathsf{Hyb}_3(F,\mathbf{x})$ and $\mathsf{Sim}_3(F,\mathbf{x})$.

Here, $\mathsf{Sim}_2(F,\mathbf{x})$ denotes the ideal-world experiment corresponding to the final simulator. The parameter $\mathbf{x}$ specifies the clients' inputs to the ideal functionality and is not initially given to the simulator; the simulator learns x_i only when $\mathcal{C}_i$ is adaptively corrupted. The experiments $\mathsf{Sim}_k(F,\mathbf{x})$ for $k \ge 3$, together with their indexed variants, are only auxiliary experiments used in the proof.

We start with the definition of $\mathsf{Hyb}_0(F,\mathbf{x})$.

- $\mathsf{Hyb}_0(F,\mathbf{x})$: This is the real-world execution of Π_F^{MPC} in the outsourced-MPC model defined in Section 3.2.

We next adapt to the present MPC setting the hybrid argument from Section 4.5. We use the same corruption and deletion notation, counting only computing-party events before successful finalization. For every corruption index $\tau \in [n]$ that occurs, let $i_\tau^{(C)}$ denote the corrupted party; for every corrupt-then-delete index $\tau' \in [n]$ that occurs, let $i_{\tau'}^{(D)}$ denote the deleted party. Honest deletions do not advance the latter counter. Put $\varphi(i) = 1$ if corrupting $\mathcal{P}_i$ delivers quantum shares, and put $\varphi(i) = 0$ otherwise. Define $\eta(\tau) := \max\{0, \sum_{\upsilon\in[\tau]} \varphi(i_\upsilon^{(C)}) - t\}$ and $\eta(0) := 0$. Let $\hat{\tau}$ be the corruption index at which the t-th quantum-share delivery occurs, or n if fewer than t such deliveries occur. All party indices refer only to events that actually occur. Adaptive client corruptions are tracked separately by $\mathcal{CC}$. They and post-finalization computing-party corruptions are handled by their existing interfaces and do not advance these counters.

For the purpose of the security experiment, we slightly extend the notion of rounds used in Section 5.2: we regard the input phase as the initial part of Round_1, and the output and finalization procedures as the terminal part of Round_Γ. For every $\nu \in [\Gamma]$, let $\mathsf{CD\text{-}phase}_\nu$ denote the adaptive-query point immediately before Round_ν. In particular, $\mathsf{CD\text{-}phase}_1$ occurs after the offline phase and before the clients submit their inputs. At each $\mathsf{CD\text{-}phase}_\nu$, the adversary may adaptively corrupt additional clients, subject to $|\mathcal{CC}| \le n-1$, and may issue any finite sequence of computing-party corruption and deletion queries satisfying the online t-limited conditions. Every such query is processed completely before the round begins, and no adaptive query may be issued during a round. Let $\mathsf{CD\text{-}phase}_{\Gamma+1}$ denote the post-execution corruption phase immediately after successful finalization. In this phase, every remaining uncorrupted computing party may be corrupted and reveals its residual state, and additional clients may be corrupted subject to $|\mathcal{CC}| \le n-1$, but no deletion query is issued. Because the online t-limited condition guarantees $|\mathcal{CS} \setminus \mathcal{DS}| \le t$, the corruption–deletion matching map η used in Section 4.5 is well defined for all computing-party corruptions before successful finalization.

We now define $\mathsf{Hyb}_1(F, \mathbf{x})$.

- $\mathsf{Hyb}_1(F, \mathbf{x})$: This experiment is identical to $\mathsf{Hyb}_0(F, \mathbf{x})$ except for the generation of the input masks. For every client $\mathcal{C}_i$, instead of first sampling $\Delta_i \leftarrow \mathbb{Z}_q$ and then setting $\Delta_i' := x_i - \Delta_i$, the challenger first samples $\Delta_i' \leftarrow \mathbb{Z}_q$ uniformly at random and then sets $\Delta_i := x_i - \Delta_i'$. The trusted preprocessing uses this value Δ_i when generating the corresponding quantum input-mask sharings.

Lemma 38. $\mathsf{TD}(\mathsf{Hyb}_1(F, \mathbf{x}), \mathsf{Hyb}_0(F, \mathbf{x})) = 0$.

Proof. The two experiments induce the same distribution, since changing the order of sampling Δ_i and Δ_i' does not affect their joint distribution subject to the constraint $\Delta_i + \Delta_i' = x_i$. In particular, this remains true for the complete client view revealed upon any adaptive client corruption.

We next define $\mathsf{Hyb}_2(F, \mathbf{x})$.

- $\mathsf{Hyb}_2(F, \mathbf{x})$: This experiment is identical to $\mathsf{Hyb}_1(F, \mathbf{x})$ except that the challenger obtains the protocol output from the ideal functionality and programs the classical output-mask sharings accordingly.

  Concretely, the challenger simulates the execution up to the reconstruction of the masked outputs exactly as in $\mathsf{Hyb}_1(F, \mathbf{x})$. During the input phase, every uncorrupted client submits

its input directly to the ideal functionality, while for every already corrupted client $\mathcal{C}_i$, the challenger submits x_i on its behalf, as prescribed by the ideal-world experiment. If the execution aborts before all masked outputs $\{o'_h\}_{h\in[n']}$ have been successfully reconstructed, the challenger sends ABORT to the ideal functionality and aborts. Otherwise, immediately before the output-mask opening, the challenger obtains $(o_1,\ldots,o_{n'}) = F(\mathbf{x})$ from the ideal functionality. For every $h \in [n']$, it defines $\gamma^\star_h := o'_h - o_h$ and invokes the programmable-completion simulator of Π_{ROpenSS} to complete the unexposed honest state so that $\langle\gamma_h\rangle$ opens to $\gamma^\star_h$.

If the simulated output-mask opening aborts, the challenger sends ABORT to the ideal functionality and aborts. Otherwise, it sends $\mathsf{DELIVER}$ to the ideal functionality, every honest computing party outputs $(o_1,\ldots,o_{n'})$, and the execution is successfully finalized.

Immediately after successful finalization, the experiment enters $\mathsf{CD\text{-}phase}_{\Gamma+1}$. Every remaining uncorrupted computing party is corrupted, whose residual state is revealed. These states contain neither the input masks $\{\Delta_i\}_{i\in[n]}$ nor any unmeasured quantum sharing.

Lemma 39. $\mathsf{TD}\big(\mathsf{Hyb}_2(F,\mathbf{x}),\mathsf{Hyb}_1(F,\mathbf{x})\big) \le \mathrm{negl}(\lambda)$.

Proof. By the correctness invariant of the protocol and Lemma 36, every quantum reconstruction either returns its prescribed masked value or aborts. In particular, whenever all masked outputs are successfully reconstructed, $o'_h = o_h + \gamma_h$ and hence $\gamma^\star_h = \gamma_h$ for every $h \in [n']$. By the statistical robust-opening property of Π_{ROpenSS} and a union bound over the n' output masks, the probability that any non-aborting classical opening in $\mathsf{Hyb}_1(F,\mathbf{x})$ returns a value different from its original mask is at most $\mathrm{negl}(\lambda)$. Outside this bad event, $\mathsf{Hyb}_1(F,\mathbf{x})$ either aborts or outputs $F(\mathbf{x})$.

Before the output-mask opening, the online honest-quorum condition ensures that at least one computing party remains uncorrupted. Since $\gamma^\star_h = \gamma_h$, the statistical programmable-completion property of Π_{ROpenSS} allows the challenger to replace the unexposed honest state by a statistically indistinguishable completion for the same shared value, conditioned on the previously exposed adversarial view. A hybrid argument over the n' output masks gives a total statistical error of at most $\mathrm{negl}(\lambda)$, including the completed states and the opening executions.

Finally, after successful finalization, both experiments reveal the residual states of all computing parties, including the classical masks retained from deleted or measured quantum sharings. The completed classical-sharing states are covered by the preceding comparison, and all other residual states are generated identically. Combining the robust-opening and programmable-completion errors therefore gives $\mathsf{TD}\big(\mathsf{Hyb}_2(F,\mathbf{x}),\mathsf{Hyb}_1(F,\mathbf{x})\big) \le \mathrm{negl}(\lambda)$. □

Note that in $\mathsf{Hyb}_2(F,\mathbf{x})$, the prescribed values reconstructed from the quantum secret-sharing batches form a uniformly distributed vector. It remains to replace the secret values underlying these batches by independent uniform values.

- $\mathsf{Sim}_2(F,\mathbf{x})$: This experiment is identical to $\mathsf{Hyb}_2(F,\mathbf{x})$ except that the secret value of each quantum sharing is replaced by a uniformly random value. Concretely, for every $\kappa \in [\Gamma]$ and every $\iota_\kappa \in [M_\kappa]$, instead of setting $\mathsf{val}_{\kappa,\iota_\kappa}$ according to Eqn (22), the simulator samples $\mathsf{val}_{\kappa,\iota_\kappa} \leftarrow \mathbb{Z}_q$. Note that, compared with $\mathsf{Hyb}_2(F,\mathbf{x})$, the computation of $\Delta_i = x_i - \Delta'_i$ is postponed from preprocessing until $\mathcal{C}_i$ is corrupted, at which point $\mathcal{S}_2$ obtains x_i from the ideal-world corruption interface.

This is the ideal-world experiment induced by a simulator $\mathcal{S}_2$ that internally runs $\mathcal{A}$ and emulates all uncorrupted computing parties. For every client $\mathcal{C}_i$, $\mathcal{S}_2$ samples its public input message $\Delta_i' \leftarrow \mathbb{Z}_q$ uniformly without knowing x_i. Unlike in $\mathsf{Hyb}_2(F, \mathbf{x})$, the simulator does not need to determine $\Delta_i = x_i - \Delta_i'$ during preprocessing, since the corresponding quantum-sharing secrets are replaced by independent uniform values. Whenever $\mathcal{C}_i$ is adaptively corrupted, $\mathcal{S}_2$ obtains x_i from the ideal-world corruption interface, sets $\Delta_i := x_i - \Delta_i'$, and provides the corresponding simulated client view to $\mathcal{A}$. During the input phase, every uncorrupted client submits x_i directly to the ideal functionality, while $\mathcal{S}_2$ submits x_i on behalf of every already corrupted client. It replaces every pre-evaluation secret value underlying the quantum secret-sharing batches by an independent uniform field element, answers the adversary's computing-party corruption and deletion queries using the simulated states, reproduces every protocol abort caused by the adversary, and obtains the prescribed output only from the ideal functionality. After successful finalization, it simulates the residual states revealed by the post-execution corruption of all remaining computing parties and continues to answer adaptive client-corruption queries in the same manner.

Our goal is to show the following lemma.

Lemma 40. $\mathsf{TD}\big(\mathsf{Hyb}_2(F, \mathbf{x}), \mathsf{Sim}_2(F, \mathbf{x})\big) = 2^{-\Omega(\lambda)}$.

Adaptive client corruptions are answered using the interfaces already specified in the two experiments. For fixed clients' inputs and public masked input messages, these responses disclose the same client-side values. We retain these values as common side information throughout the following joint-state argument. This argument compares the adversary's views together with the client-side disclosures and retained classical masks, including adaptive client-corruption queries, the prescribed reconstructions, and post-execution computing-party corruptions.

We next define $\mathsf{Hyb}_3(F, \mathbf{x})$ and $\mathsf{Sim}_3(F, \mathbf{x})$.

- $\mathsf{Hyb}_3(F, \mathbf{x})$ (resp. $\mathsf{Sim}_3(F, \mathbf{x})$): This experiment is identical to $\mathsf{Hyb}_2(F, \mathbf{x})$ (resp. $\mathsf{Sim}_2(F, \mathbf{x})$) except that the challenger generates each quantum secret-sharing batch by first sampling its description after evaluation and then applying the inverse evaluation transformation.

  For every $\kappa \in [\Gamma]$, let $\mathbf{val}_\kappa := (\mathsf{val}_{\kappa,1}, \ldots, \mathsf{val}_{\kappa,M_\kappa})^T$ denote the pre-evaluation secret-value vector in the corresponding experiment.

  Let $\boldsymbol{f}^{(\kappa)}(z) := (f_1^{(\kappa)}(z), \ldots, f_{M_\kappa}^{(\kappa)}(z))^T$ and $\boldsymbol{\sigma}^{(\kappa)} := (\sigma_1^{(\kappa)}, \ldots, \sigma_{M_\kappa}^{(\kappa)})^T$ denote the polynomial-evaluation vector and classical-mask vector before evaluation, and define $\widetilde{\boldsymbol{f}}^{(\kappa)}(z)$ and $\widetilde{\boldsymbol{\sigma}}^{(\kappa)}$ analogously after evaluation. Fix the initially sampled client-side values and the pre-evaluation secret vectors. Run the prescribed abstract computation, using the value encoded by each designated sharing whenever a reconstruction is required. This recursively specifies the reconstructed values for every batch, including batches not reached on an aborting execution. For every $\kappa \in [\Gamma]$, let $\mathbf{V}_\kappa \in \mathbb{Z}_q^{M_\kappa \times M_\kappa}$ be the resulting invertible linear transformation on polynomial-evaluation vectors, and let $\mathbf{off}_\kappa \in \mathbb{Z}_q^{M_\kappa}$ be the resulting offset from public scalar-addition operations. These transformations depend only on the public input messages and preceding prescribed reconstructed values. In particular, they do not depend on the classical masks $\boldsymbol{\sigma}^{(\kappa)}$, the verification setup, or the nonconstant polynomial coefficients. Thus, for every evaluation point z,

$$\widetilde{\boldsymbol{f}}^{(\kappa)}(z) = \mathbf{V}_\kappa \boldsymbol{f}^{(\kappa)}(z), \qquad \widetilde{\boldsymbol{\sigma}}^{(\kappa)} = \mathbf{V}_\kappa \boldsymbol{\sigma}^{(\kappa)} + \mathbf{off}_\kappa.$$

Define the post-evaluation secret-value vector by $\widetilde{\mathbf{val}}_\kappa := \mathbf{V}_\kappa \mathbf{val}_\kappa + \mathbf{off}_\kappa$. The challenger samples $\widetilde{\boldsymbol{\sigma}}^{(\kappa)} \leftarrow \mathbb{Z}_q^{M_\kappa}$ uniformly at random, sets $\widetilde{\boldsymbol{f}}^{(\kappa)}(0) := \widetilde{\mathbf{val}}_\kappa - \widetilde{\boldsymbol{\sigma}}^{(\kappa)}$, and generates the remaining post-evaluation polynomial evaluations consistently with these constant terms. It then defines

$$\boldsymbol{\sigma}^{(\kappa)} := \mathbf{V}_\kappa^{-1}\big(\widetilde{\boldsymbol{\sigma}}^{(\kappa)} - \mathbf{off}_\kappa\big), \qquad \boldsymbol{f}^{(\kappa)}(z) := \mathbf{V}_\kappa^{-1}\widetilde{\boldsymbol{f}}^{(\kappa)}(z).$$

The executions of Π_{ROpenSS} associated with $\{\langle\gamma_h\rangle\}_{h\in[n']}$ remain unchanged.

Lemma 41. $\mathsf{TD}\big(\mathsf{Hyb}_2(F,\mathbf{x}), \mathsf{Hyb}_3(F,\mathbf{x})\big) = 0$ *and* $\mathsf{TD}\big(\mathsf{Sim}_2(F,\mathbf{x}), \mathsf{Sim}_3(F,\mathbf{x})\big) = 0$.

Proof. Fix the initially sampled client-side values and pre-evaluation secret vectors. The prescribed abstract computation determines all $\mathbf{V}_\kappa$ and $\mathbf{off}_\kappa$. This description is valid for arbitrary initial secret vectors: linear evaluation preserves the degree bound on each honest sharing. If a reconstruction accepts a polynomial $\widehat{f}$, then, for each honest undeleted party, $\widehat{f}$ agrees with the polynomial f encoded by its data registers at at least $m - 2n\ell$ points. The honest-quorum condition and $d < (t+1)(m-2n\ell)$ imply that $f - \widehat{f}$ has more than d distinct roots, so $f = \widehat{f}$. Inductively, each reconstruction therefore returns its prescribed encoded value or aborts, and all transformations applied before an abort are the prescribed ones. This argument does not require the initial secrets to encode the real MPC preprocessing values.

For each batch, the map $(\boldsymbol{f}^{(\kappa)}, \boldsymbol{\sigma}^{(\kappa)}) \mapsto (\mathbf{V}_\kappa \boldsymbol{f}^{(\kappa)}, \mathbf{V}_\kappa \boldsymbol{\sigma}^{(\kappa)} + \mathbf{off}_\kappa)$ is a bijection between the pre-evaluation and post-evaluation descriptions of the same sharing. Sampling the post-evaluation description first and applying its inverse therefore preserves the complete joint distribution. Averaging over the fixed values proves both equalities. □

We next define $\mathsf{Hyb}_4(F,\mathbf{x})$ and $\mathsf{Sim}_4(F,\mathbf{x})$.

- $\mathsf{Hyb}_4(F,\mathbf{x})$ (resp. $\mathsf{Sim}_4(F,\mathbf{x})$): it is the same as $\mathsf{Hyb}_3(F,\mathbf{x})$ (resp. $\mathsf{Sim}_3(F,\mathbf{x})$) except that whenever an honest party is instructed to delete its share, the challenger $\mathcal{C}$ does not generate the corresponding deletion certificate by honestly measuring the share. Instead, $\mathcal{C}$ simulates the certificate from $\mathcal{A}$'s view.

  More concretely, in $\mathsf{CD\text{-}phase}_\nu$ for $\nu \in [\Gamma]$, when an honest $\mathcal{P}_{i_1}$ is instructed to delete its share ($\varphi(i_1) = 0$), for each $\kappa \in [\nu, \Gamma]$, $\iota_\kappa \in [M_\kappa]$, the challenger $\mathcal{C}$ generates $\mathsf{cert}^{(\kappa)}_{i_1,\iota_\kappa}$ as follows:

  1. For each $j \in \mathcal{J}_{i_1}^{\times(\kappa)}$, let $i_2 \in [n]$ be the unique index such that $j \in \mathsf{mvk}^{(\kappa)}_{i_2,i_1}$. We let $\mathsf{cert}^{(\kappa)}_{i_1,\iota_\kappa}[j] = \mathsf{vk}^{(\kappa)}_{i_2,i_1,\iota_\kappa}[j]$.
  2. For each $j \in \mathcal{J}_{i_1}^{\checkmark(\kappa)}$, sample $\mathsf{cert}^{(\kappa)}_{i_1,\iota_\kappa}[j] \leftarrow \mathbb{Z}_q$ uniformly at random.

  In the following scheduled deletion procedures, vk denotes the current verification key after the prescribed evaluation. In the evaluation phase, in the deleting process before reconstructing $u'_{\mathsf{cur}}, v'_{\mathsf{cur}}$ in $\mathsf{Round}_{\mathsf{cur}}$, for each undeleted honest party $\mathcal{P}_{i_1}$, each $\iota \in [M_{\mathsf{cur}} - 2]$, the challenger $\mathcal{C}$ generates $\mathsf{cert}^{(\mathsf{cur})}_{i_1,\iota}$ as follows.

  - For each $j \in \mathcal{J}_{i_1}^{\times(\mathsf{cur})}$, let $i_2 \in [n]$ be the unique index such that $j \in \mathsf{mvk}^{(\mathsf{cur})}_{i_2,i_1}$. We let $\mathsf{cert}^{(\mathsf{cur})}_{i_1,\iota}[j] = \mathsf{vk}^{(\mathsf{cur})}_{i_2,i_1,\iota}[j]$.
  - For each $j \in \mathcal{J}_{i_1}^{\checkmark(\mathsf{cur})}$, sample $\mathsf{cert}^{(\mathsf{cur})}_{i_1,\iota}[j] \leftarrow \mathbb{Z}_q$ uniformly at random.

In the output phase, in the deleting process before reconstructing $\{o'_h\}_{h\in[n']}$, for each undeleted honest party $\mathcal{P}_{i_1}$, each $\iota \in [N+n]$, the challenger $\mathcal{C}$ generates $\mathsf{cert}^{(\Gamma)}_{i_1,\iota}$ as follows.

- For each $j \in \mathcal{J}_{i_1}^{\times(\Gamma)}$, let $i_2 \in [n]$ be the unique index such that $j \in \mathsf{mvk}^{(\Gamma)}_{i_2,i_1}$. We let $\mathsf{cert}^{(\Gamma)}_{i_1,\iota}[j] = \mathsf{vk}^{(\Gamma)}_{i_2,i_1,\iota}[j]$.
- For each $j \in \mathcal{J}_{i_1}^{\checkmark(\Gamma)}$, sample $\mathsf{cert}^{(\Gamma)}_{i_1,\iota}[j] \leftarrow \mathbb{Z}_q$ uniformly at random.

Lemma 42. $\mathsf{TD}(\mathsf{Hyb}_3(F,\mathbf{x}), \mathsf{Hyb}_4(F,\mathbf{x})) = 0$, $\mathsf{TD}(\mathsf{Sim}_3(F,\mathbf{x}), \mathsf{Sim}_4(F,\mathbf{x})) = 0$.

Proof. We omit the proof since it is almost the same as that of Lemma 28. □

- $\mathsf{Hyb}_5(F,\mathbf{x})$ (resp. $\mathsf{Sim}_5(F,\mathbf{x})$): It is the same as $\mathsf{Hyb}_4(F,\mathbf{x})$ (resp. $\mathsf{Sim}_4(F,\mathbf{x})$) except that the challenger $\mathcal{C}$ generates the shares lazily using polynomial interpolation.

  Concretely, in the offline phase, instead of honestly invoking the Split algorithm of each $\Pi^{(\kappa)}$, $\mathcal{C}$ goes through the following process.

  1. For every $\kappa \in [\Gamma]$, let $\mathbf{val}_\kappa, \widetilde{\mathbf{val}}_\kappa, \mathbf{V}_\kappa, \mathbf{off}_\kappa, \boldsymbol{\sigma}^{(\kappa)}, \widetilde{\boldsymbol{\sigma}}^{(\kappa)}$ be generated exactly as in $\mathsf{Hyb}_4(F,\mathbf{x})$ (resp. $\mathsf{Sim}_4(F,\mathbf{x})$).
  2. For every $\kappa \in [\Gamma]$, set $\widetilde{\boldsymbol{f}}^{(\kappa)}(0) := \widetilde{\mathbf{val}}_\kappa - \widetilde{\boldsymbol{\sigma}}^{(\kappa)}$. Define $\boldsymbol{f}^{(\kappa)}(0) := \mathbf{V}_\kappa^{-1}\widetilde{\boldsymbol{f}}^{(\kappa)}(0)$. We have $\boldsymbol{f}^{(\kappa)}(0) + \boldsymbol{\sigma}^{(\kappa)} = \mathbf{val}_\kappa$.
  3. For every $\kappa \in [\Gamma]$, every $\iota_\kappa \in [M_\kappa]$, every $i_1, i_2 \in [n]$, and every $j \in \mathsf{mvk}^{(\kappa)}_{i_2,i_1}$, sample $r^{(\kappa)}_{i_1,j,\iota_\kappa} \leftarrow \mathbb{Z}_q$, and set $|\psi^{(\kappa)}_{i_1,j,\iota_\kappa}\rangle := \mathsf{QFT}|r^{(\kappa)}_{i_1,j,\iota_\kappa}\rangle$. Let $\mathsf{vk}^{(\kappa)}_{i_2,i_1,\iota_\kappa} := \{(j, r^{(\kappa)}_{i_1,j,\iota_\kappa})\}_{j\in\mathsf{mvk}^{(\kappa)}_{i_2,i_1}}$.
  4. For every $\kappa \in [\Gamma]$ and every $i_1 \in [n]$, randomly partition $\mathcal{J}_{i_1}^{\checkmark(\kappa)}$ into two sets $\mathcal{J}_{i_1}^{L(\kappa)}$ and $\mathcal{J}_{i_1}^{R(\kappa)}$ such that $|\mathcal{J}_{i_1}^{L(\kappa)}| = m'_L$ and $|\mathcal{J}_{i_1}^{R(\kappa)}| = m'_R$. For every $j \in \mathcal{J}_{i_1}^{L(\kappa)}$ and every $\iota_\kappa \in [M_\kappa]$, sample $\tilde{f}^{(\kappa)}_{\iota_\kappa}((i_1-1)m+j) \leftarrow \mathbb{Z}_q$. Let

  $$\widetilde{\boldsymbol{f}}^{(\kappa)}((i_1-1)m+j) := \big(\tilde{f}^{(\kappa)}_1((i_1-1)m+j), \ldots, \tilde{f}^{(\kappa)}_{M_\kappa}((i_1-1)m+j)\big)^T$$

  and define $\boldsymbol{f}^{(\kappa)}((i_1-1)m+j) := \mathbf{V}_\kappa^{-1}\widetilde{\boldsymbol{f}}^{(\kappa)}((i_1-1)m+j)$. Set $|\psi^{(\kappa)}_{i_1,j,\iota_\kappa}\rangle := |f^{(\kappa)}_{\iota_\kappa}((i_1-1)m+j)\rangle$ for every $\iota_\kappa \in [M_\kappa]$. For each batch κ, initialize the set of nonzero source evaluation points as

  $$\mathcal{K}^{(\kappa)} := \bigcup_{i\in[n]} \{(i-1)m+j\}_{j\in\mathcal{J}_i^{L(\kappa)}}. \tag{27}$$

  This set gains the R points of each of the first t quantum-share deliveries involving this batch, as specified below. Once its size reaches $d = nm'_L + tm'_R$, it remains fixed. Deleting a party never removes or replaces a source evaluation; in particular, all L sources are retained.

In $\mathsf{CD\text{-}phase}_\nu$ for $\nu \in [\Gamma+1]$, when $\mathcal{A}$ corrupts a party $\mathcal{P}_{i_1}$, the challenger does the following:

– The challenger computes the current masks and current verification keys of $\mathcal{P}_{i_1}$ from the offline values

$$\left(\left\{\mathsf{mvk}^{(\kappa)}_{i_1,i_2}\right\}_{i_2\in[n],\,\kappa\in[\Gamma]},\left\{\mathsf{vk}^{(\kappa)}_{i_1,i_2,\iota_\kappa}\right\}_{i_2\in[n],\,\kappa\in[\Gamma],\,\iota_\kappa\in[M_\kappa]},\{\sigma^{(\kappa)}_{\iota_\kappa}\}_{\kappa\in[\Gamma],\iota_\kappa\in[M_\kappa]}\right)$$

and the public transcript, by applying the prescribed classical updates up to the time of corruption. It sends the resulting current state, including the master verification keys, together with all classical local randomness, received broadcast messages, and public transcript information determined up to that time.

– If $\nu\in[\Gamma]$ and $\mathcal{P}_{i_1}$ is an undeleted honest party ($i_1=i^{(C)}_\tau$ for some τ that $\varphi(i^{(C)}_\tau)=1$), the challenger $\mathcal{C}$ additionally does the following.

  * If $\tau\in[1,\hat\tau]$, then, for every $\kappa\in[\nu,\Gamma]$ and every $j\in\mathcal{J}^{R(\kappa)}_{i^{(C)}_\tau}$, sample $\widetilde{\boldsymbol{f}}^{(\kappa)}((i^{(C)}_\tau-1)m+j)\leftarrow\mathbb{Z}_q^{M_\kappa}$ uniformly and independently, and set $\boldsymbol{f}^{(\kappa)}((i^{(C)}_\tau-1)m+j):=\mathbf{V}_\kappa^{-1}\widetilde{\boldsymbol{f}}^{(\kappa)}((i^{(C)}_\tau-1)m+j)$. For every $\iota_\kappa\in[M_\kappa]$, set $|\psi^{(\kappa)}_{i^{(C)}_\tau,j,\iota_\kappa}\rangle:=|f^{(\kappa)}_{\iota_\kappa}((i^{(C)}_\tau-1)m+j)\rangle$. For each $\kappa\in[\nu,\Gamma]$, add $\{(i^{(C)}_\tau-1)m+j\}_{j\in\mathcal{J}^{R(\kappa)}_{i^{(C)}_\tau}}$ to $\mathcal{K}^{(\kappa)}$. For every $\kappa\in[\nu,\Gamma]$ and $\iota_\kappa\in[M_\kappa]$, let $\mathsf{sh}^{(\kappa)}_{i^{(C)}_\tau,\iota_\kappa}:=(|\boldsymbol{\psi}^{(\kappa)}_{i^{(C)}_\tau,\iota_\kappa}\rangle,\sigma^{(\kappa)}_{\iota_\kappa})$. The challenger $\mathcal{C}$ replaces each mask component by the corresponding current mask computed above, and then gives $\{\mathsf{sh}^{(\kappa)}_{i^{(C)}_\tau,\iota_\kappa}\}_{\kappa\in[\nu,\Gamma],\,\iota_\kappa\in[M_\kappa]}$ to $\mathcal{A}$.

  * If $\tau\in(\hat\tau,n]$, then, for every $\kappa\in[\nu,\Gamma]$, every $\iota_\kappa\in[M_\kappa]$, and every $j\in\mathcal{J}^{R(\kappa)}_{i^{(C)}_\tau}$, interpolate $\widetilde{f}^{(\kappa)}_{\iota_\kappa}((i^{(C)}_\tau-1)m+j)$ from $\{\widetilde{f}^{(\kappa)}_{\iota_\kappa}(k)\}_{k\in\{0\}\cup\mathcal{K}^{(\kappa)}}$ using Lemma 7. Set $\boldsymbol{f}^{(\kappa)}((i^{(C)}_\tau-1)m+j):=\mathbf{V}_\kappa^{-1}\widetilde{\boldsymbol{f}}^{(\kappa)}((i^{(C)}_\tau-1)m+j)$ and $|\psi^{(\kappa)}_{i^{(C)}_\tau,j,\iota_\kappa}\rangle:=|f^{(\kappa)}_{\iota_\kappa}((i^{(C)}_\tau-1)m+j)\rangle$. For every $\kappa\in[\nu,\Gamma]$ and $\iota_\kappa\in[M_\kappa]$, let $\mathsf{sh}^{(\kappa)}_{i^{(C)}_\tau,\iota_\kappa}:=(|\boldsymbol{\psi}^{(\kappa)}_{i^{(C)}_\tau,\iota_\kappa}\rangle,\sigma^{(\kappa)}_{\iota_\kappa})$. The challenger $\mathcal{C}$ replaces each mask component by the corresponding current mask computed above, and then gives $\{\mathsf{sh}^{(\kappa)}_{i^{(C)}_\tau,\iota_\kappa}\}_{\kappa\in[\nu,\Gamma],\,\iota_\kappa\in[M_\kappa]}$ to $\mathcal{A}$.

- In the evaluation phase, when reconstructing $u'_{\mathsf{cur}},v'_{\mathsf{cur}}$ in $\mathsf{Round}_{\mathsf{cur}}$, for each undeleted honest party $\mathcal{P}_{i_1}$, the challenger $\mathcal{C}$ simulates the measurement results as follows. Denote the auxiliary wires by $\mathsf{AUX}_{\iota_3},\mathsf{AUX}_{\iota_4}$. Before generating these messages, complete the source evaluations, if necessary, once for each reconstructed coordinate.[5] The interpolations below use these completed sources; if the t-th quantum-share delivery has already occurred, they are exactly $\{0\}\cup\mathcal{K}^{(\mathsf{cur})}$. For each $j\in\mathcal{J}^{\times(\mathsf{cur})}_{i_1}$, sample $z^{(\mathsf{cur})}_{i_1,\iota_3}[j],z^{(\mathsf{cur})}_{i_1,\iota_4}[j]\leftarrow\mathbb{Z}_q$ uniformly at random. For each $j\in\mathcal{J}^{L(\mathsf{cur})}_{i_1}$, let $z^{(\mathsf{cur})}_{i_1,\iota_3}[j]=\tilde f^{(\mathsf{cur})}_{\iota_3}((i_1-1)m+j),z^{(\mathsf{cur})}_{i_1,\iota_4}[j]=\tilde f^{(\mathsf{cur})}_{\iota_4}((i_1-1)m+j)$. For $j\in\mathcal{J}^{R(\mathsf{cur})}_{i_1}$, interpolate $z^{(\mathsf{cur})}_{i_1,\iota_3}[j]$, $z^{(\mathsf{cur})}_{i_1,\iota_4}[j]$ from the corresponding fixed source evaluations, completed as just described. Broadcast $\mathbf{z}^{(\mathsf{cur})}_{i_1,\iota_3},\mathbf{z}^{(\mathsf{cur})}_{i_1,\iota_4}$ for $\mathcal{P}_{i_1}$.

[5] If fewer than d independent nonzero source evaluations have been fixed, choose distinct unused nonzero evaluation points and independently sample their values until there are d such source evaluations. Together with the fixed value at 0, these determine a unique degree-at-most-d polynomial. All existing source values are retained. This completion is performed only for the coordinates actually reconstructed; the other coordinates are unchanged. The output sharings are handled in the same way.

- In the output phase, when reconstructing $\{o'_h\}_{h\in[n']}$, for each undeleted honest party $\mathcal{P}_{i_1}$, the challenger $\mathcal{C}$ simulates the measurement results as follows. For every $h \in [n']$, let $\iota_h := n+N+h$ and complete its source evaluations as above. For each $j \in \mathcal{J}_{i_1}^{\times(\Gamma)}$, sample $z^{(\Gamma)}_{i_1,\iota_h}[j] \leftarrow \mathbb{Z}_q$ uniformly at random. For each $j \in \mathcal{J}_{i_1}^{L(\Gamma)}$, let $z^{(\Gamma)}_{i_1,\iota_h}[j] = \tilde{f}^{(\Gamma)}_{\iota_h}((i_1-1)m+j)$. For $j \in \mathcal{J}_{i_1}^{R(\Gamma)}$, interpolate $z^{(\Gamma)}_{i_1,\iota_h}[j]$ from the fixed source evaluations for coordinate ι_h, completed if necessary. Broadcast $\mathbf{z}^{(\Gamma)}_{i_1,\iota_h}$ for $\mathcal{P}_{i_1}$.
- All remaining aspects of the experiment, if not mentioned, are identical to those in $\mathsf{Hyb}_4(F,\mathbf{x})$ (resp. $\mathsf{Sim}_4(F,\mathbf{x})$).

Lemma 43. $\mathsf{TD}(\mathsf{Hyb}_4(F,\mathbf{x}),\mathsf{Hyb}_5(F,\mathbf{x})) = 0$, $\mathsf{TD}(\mathsf{Sim}_4(F,\mathbf{x}),\mathsf{Sim}_5(F,\mathbf{x})) = 0$.

Proof. This proof is similar to that of Lemma 29. For each batch, the challenger first samples the polynomial evaluations and classical masks in the post-evaluation coordinate system and then applies the inverse transformation

$$\boldsymbol{f}^{(\kappa)} = \mathbf{V}_\kappa^{-1}\widetilde{\boldsymbol{f}}^{(\kappa)}, \qquad \boldsymbol{\sigma}^{(\kappa)} = \mathbf{V}_\kappa^{-1}\big(\widetilde{\boldsymbol{\sigma}}^{(\kappa)} - \mathbf{off}_\kappa\big).$$

Since this is a bijection, it does not change the joint distribution. For each coordinate with its value at 0 fixed, evaluations at any d distinct nonzero points are independent and uniform. The deferred sampling above therefore generates the same uniformly random degree-at-most-d polynomial. Once these source values have been fixed, every later interpolation uses the same polynomial. The reconstruction completion rule is exactly the same deferred sampling applied to the remaining independent source values. Hence the two experiments are identically distributed. □

- $\mathsf{Hyb}_6(F,\mathbf{x})$ (resp. $\mathsf{Sim}_6(F,\mathbf{x})$): This game is the same as $\mathsf{Hyb}_5(F,\mathbf{x})$ (resp. $\mathsf{Sim}_5(F,\mathbf{x})$) except that $\mathcal{C}$ purifies the quantum shares.

  Concretely, for every $i_1 \in [n]$, $\kappa \in [\Gamma]$ and $\iota_\kappa \in [M_\kappa]$, the challenger $\mathcal{C}$ prepares a purification of $|\boldsymbol{\psi}^{(\kappa)}_{i_1,\iota_\kappa}\rangle$ using two compound registers

  $$\mathsf{C}^{(\kappa)}_{i_1,\iota_\kappa} := (\mathsf{C}^{(\kappa)}_{i_1,1,\iota_\kappa},\dots,\mathsf{C}^{(\kappa)}_{i_1,m,\iota_\kappa}) \quad \text{and} \quad \mathsf{A}^{(\kappa)}_{i_1,\iota_\kappa} := (\mathsf{A}^{(\kappa)}_{i_1,1,\iota_\kappa},\dots,\mathsf{A}^{(\kappa)}_{i_1,m,\iota_\kappa}).$$

  The register $\mathsf{A}^{(\kappa)}_{i_1,\iota_\kappa}$ plays the role of the share if it is delivered to the adversary, replacing the original register $|\boldsymbol{\psi}^{(\kappa)}_{i_1,\iota_\kappa}\rangle$, while $\mathsf{C}^{(\kappa)}_{i_1,\iota_\kappa}$ is retained by $\mathcal{C}$ as the purifying register. An undelivered A register also remains private to $\mathcal{C}$. For every $\kappa \in [\Gamma]$ and $\iota_\kappa \in [M_\kappa]$, define

  $$\mathsf{C}^{\mathsf{base},(\kappa)}_{\iota_\kappa} := \big(\mathsf{C}^{(\kappa)}_{\lceil k/m\rceil,((k-1) \bmod m)+1,\iota_\kappa}\big)_{k\in\mathcal{K}^{(\kappa)}},$$

  with the registers listed in a fixed order. Here $\mathcal{K}^{(\kappa)}$ evolves as in Hyb_5 (resp. Sim_5), and $\mathsf{C}^{\mathsf{base},(\kappa)}_{\iota_\kappa}$ consists of the corresponding private data registers originally prepared as halves of fresh EPR pairs. The register $\mathsf{C}^{(\kappa)}_{0,\iota_\kappa}$ is kept separately. The joint states are prepared as follows.

  1. For every $\kappa \in [\Gamma]$, first generate $\widetilde{\mathbf{val}}_\kappa$, $\widetilde{\boldsymbol{\sigma}}^{(\kappa)}$, $\boldsymbol{\sigma}^{(\kappa)}$, $\widetilde{\boldsymbol{f}}^{(\kappa)}(0)$, and $\boldsymbol{f}^{(\kappa)}(0)$ exactly as in $\mathsf{Hyb}_5(F,\mathbf{x})$ (resp. $\mathsf{Sim}_5(F,\mathbf{x})$). For every $\iota_\kappa \in [M_\kappa]$, prepare a private register $\mathsf{C}^{(\kappa)}_{0,\iota_\kappa}$ in the state $|\widetilde{f}^{(\kappa)}_{\iota_\kappa}(0)\rangle$. For every $i_1 \in [n]$, every $\iota_\kappa \in [M_\kappa]$, and every $j \in [m]$, initialize $\mathsf{C}^{(\kappa)}_{i_1,j,\iota_\kappa}$ and $\mathsf{A}^{(\kappa)}_{i_1,j,\iota_\kappa}$ to $|0\rangle$.

2. For every $\kappa \in [\Gamma]$, every $\iota_\kappa \in [M_\kappa]$, every $i_1, i_2 \in [n]$, and every $j \in \mathsf{mvk}^{(\kappa)}_{i_2,i_1}$, prepare the state

$$q^{-1/2} \sum_{x \in \mathbb{Z}_q} |x\rangle_{\mathsf{C}^{(\kappa)}_{i_1,j,\iota_\kappa}} \otimes |x\rangle_{\mathsf{A}^{(\kappa)}_{i_1,j,\iota_\kappa}} = q^{-1/2} \sum_{x \in \mathbb{Z}_q} \mathsf{QFT}^{-1}|x\rangle_{\mathsf{C}^{(\kappa)}_{i_1,j,\iota_\kappa}} \otimes \mathsf{QFT}|x\rangle_{\mathsf{A}^{(\kappa)}_{i_1,j,\iota_\kappa}}$$

on the two registers $\mathsf{C}^{(\kappa)}_{i_1,j,\iota_\kappa}$ and $\mathsf{A}^{(\kappa)}_{i_1,j,\iota_\kappa}$. Next, measure $\mathsf{C}^{(\kappa)}_{i_1,j,\iota_\kappa}$ on the inverse-QFT basis, and let the outcome be $r^{(\kappa)}_{i_1,j,\iota_\kappa}$. Define $\mathsf{vk}^{(\kappa)}_{i_2,i_1,\iota_\kappa} := \{(j, r^{(\kappa)}_{i_1,j,\iota_\kappa})\}_{j \in \mathsf{mvk}^{(\kappa)}_{i_2,i_1}}$.

3. For every $\kappa \in [\Gamma]$ and every $i_1 \in [n]$, choose the subsets $\mathcal{J}^{L(\kappa)}_{i_1}, \mathcal{J}^{R(\kappa)}_{i_1}$ exactly as in $\mathsf{Hyb}_5(F, \mathbf{x})$, using the same partition for every sharing coordinate. For every $\iota_\kappa \in [M_\kappa]$ and every $j \in \mathcal{J}^{L(\kappa)}_{i_1}$, prepare the state

$$q^{-1/2} \sum_{x \in \mathbb{Z}_q} |x\rangle_{\mathsf{C}^{(\kappa)}_{i_1,j,\iota_\kappa}} \otimes |x\rangle_{\mathsf{A}^{(\kappa)}_{i_1,j,\iota_\kappa}}$$

on the two registers $\mathsf{C}^{(\kappa)}_{i_1,j,\iota_\kappa}$ and $\mathsf{A}^{(\kappa)}_{i_1,j,\iota_\kappa}$.

4. Let $\mathbf{V}_\kappa$ and $\mathbf{V}_\kappa^{-1}$ be defined as in $\mathsf{Hyb}_3(F, \mathbf{x})$ (resp. $\mathsf{Sim}_3(F, \mathbf{x})$). We abuse notation and let them also denote the corresponding quantum operations on computational-basis states, namely $\mathbf{V}_\kappa|\mathbf{a}\rangle := |\mathbf{V}_\kappa \mathbf{a}\rangle$ and $\mathbf{V}_\kappa^{-1}|\mathbf{a}\rangle := |\mathbf{V}_\kappa^{-1}\mathbf{a}\rangle$. Keep every data register that has not been delivered in post-evaluation coordinates. The inverse transformation is applied only when the complete outgoing data block is delivered, as specified below. Checking-position registers remain in their original coordinates.

In $\mathsf{CD\text{-}phase}_\nu$ for $\nu \in [\Gamma + 1]$, when $\mathcal{A}$ corrupts a party $\mathcal{P}_{i_1}$, the challenger does the following:

– The challenger computes the current masks and current verification keys of $\mathcal{P}_{i_1}$ from the offline values

$$\left(\left\{ \mathsf{mvk}^{(\kappa)}_{i_1,i_2} \right\}_{i_2 \in [n],\, \kappa \in [\Gamma]}, \left\{ \mathsf{vk}^{(\kappa)}_{i_1,i_2,\iota_\kappa} \right\}_{i_2 \in [n],\, \kappa \in [\Gamma],\, \iota_\kappa \in [M_\kappa]}, \{\sigma^{(\kappa)}_{\iota_\kappa}\}_{\kappa \in [\Gamma], \iota_\kappa \in [M_\kappa]} \right)$$

and the public transcript, by applying the prescribed classical updates up to the time of corruption. It sends the resulting current state, including the master verification keys, together with all classical local randomness, received broadcast messages, and public transcript information determined up to that time.

– If $\nu \in [\Gamma]$ and $\mathcal{P}_{i_1}$ is an undeleted honest party ($i_1 = i^{(C)}_\tau$ for some τ that $\varphi(i^{(C)}_\tau) = 1$), the challenger $\mathcal{C}$ additionally does the following. The offline masks $\boldsymbol{\sigma}^{(\kappa)}$ remain fixed and are never resampled; their current values are computed as above, including when the corresponding quantum registers have already been deleted or measured.

  * If $\tau \in [1, \hat{\tau}]$, then, for every $\kappa \in [\nu, \Gamma]$, every $\iota_\kappa \in [M_\kappa]$, and every $j \in \mathcal{J}^{R(\kappa)}_{i^{(C)}_\tau}$, the challenger $\mathcal{C}$ prepares the state

$$q^{-1/2} \sum_{x \in \mathbb{Z}_q} |x\rangle_{\mathsf{C}^{(\kappa)}_{i^{(C)}_\tau,j,\iota_\kappa}} \otimes |x\rangle_{\mathsf{A}^{(\kappa)}_{i^{(C)}_\tau,j,\iota_\kappa}}$$

on the two registers $\mathsf{C}^{(\kappa)}_{i^{(C)}_\tau,j,\iota_\kappa}$ and $\mathsf{A}^{(\kappa)}_{i^{(C)}_\tau,j,\iota_\kappa}$.

∗ If $\tau \in (\hat{\tau}, n]$, then, for every $\kappa \in [\nu, \Gamma]$, every $\iota_\kappa \in [M_\kappa]$, and every $j \in \mathcal{J}^{R(\kappa)}_{i^{(C)}_\tau}$, run

$$\mathsf{C}^{(\kappa)}_{i^{(C)}_\tau,j,\iota_\kappa} \leftarrow \mathsf{Interpolate}_{\mathbb{Z}_q,d}\Big((i^{(C)}_\tau - 1)m + j, \{(0, \mathsf{C}^{(\kappa)}_{0,\iota_\kappa})\} \cup \Big\{\Big(k, \mathsf{C}^{(\kappa)}_{\lceil k/m\rceil,((k-1) \bmod m)+1,\iota_\kappa}\Big)\Big\}_{k\in\mathcal{K}^{(\kappa)}}\Big),$$

and copy its computational-basis content to $\mathsf{A}^{(\kappa)}_{i^{(C)}_\tau,j,\iota_\kappa}$.

In either case, for every $\kappa \in [\nu, \Gamma]$ and every $j \in \mathcal{J}^{\checkmark(\kappa)}_{i^{(C)}_\tau}$, apply $\mathbf{V}^{-1}_\kappa$ once to the complete outgoing data block $(\mathsf{A}^{(\kappa)}_{i^{(C)}_\tau,j,1}, \ldots, \mathsf{A}^{(\kappa)}_{i^{(C)}_\tau,j,M_\kappa})$. For every $\iota_\kappa \in [M_\kappa]$, let $\mathsf{sh}^{(\kappa)}_{i^{(C)}_\tau,\iota_\kappa} := (\mathsf{A}^{(\kappa)}_{i^{(C)}_\tau,\iota_\kappa}, \sigma^{(\kappa)}_{\iota_\kappa})$. The challenger $\mathcal{C}$ replaces each mask component by the corresponding current mask computed above, and then gives $\{\mathsf{sh}^{(\kappa)}_{i^{(C)}_\tau,\iota_\kappa}\}_{\kappa\in[\nu,\Gamma],\,\iota_\kappa\in[M_\kappa]}$ to $\mathcal{A}$.

- For an honest reconstruction in the evaluation or output phase, use the post-evaluation data registers of the coordinates being reconstructed. If fewer than d independent nonzero source values have been prepared, append the missing uniform source values at distinct unused nonzero evaluation points coherently, retaining all existing source registers and their correlations, including the L sources of honestly deleted parties. Concretely, for each new source point append a fresh EPR pair, retain both halves privately, and use its C register as a source for subsequent coherent interpolation. Together with $\mathsf{C}^{(\kappa)}_{0,\iota_\kappa}$, the completed source values determine a degree-at-most-d polynomial. Coherently compute the required honest data values from these sources, measure the outgoing copies in the computational basis, and generate the checking-position outcomes as in Hyb_5 (resp. Sim_5). Perform this completion only for coordinates actually reconstructed; do not modify the coordinates deleted before reconstruction.

  The challenger retains every private purifying register until the end of the experiment, including never-delivered A registers and data registers of honestly deleted parties. Simulating an honest deletion certificate does not measure or discard these private registers. After a batch has completed its scheduled deletion and reconstruction, freeze all private data registers of its deleted coordinates; no later operation acts on them. These are challenger-private registers and are never disclosed by subsequent computing-party corruptions.

- All remaining aspects of the experiment, if not mentioned, are identical to those in $\mathsf{Hyb}_5(F, \mathbf{x})$ (resp. $\mathsf{Sim}_5(F, \mathbf{x})$).

Lemma 44. $\mathsf{TD}(\mathsf{Hyb}_5(F, \mathbf{x}), \mathsf{Hyb}_6(F, \mathbf{x})) = 0$, $\mathsf{TD}(\mathsf{Sim}_5(F, \mathbf{x}), \mathsf{Sim}_6(F, \mathbf{x})) = 0$.

Proof. The proof is similar to that of Lemma 30. The classical variables are generated identically in the two experiments. Preparing fresh EPR pairs and retaining their private halves purifies the uniform source sampling in Hyb_5 (resp. Sim_5), and coherent interpolation gives a purification of the same polynomial evaluations. The inverse transformation on an undelivered data block acts only on challenger-private registers. It may be postponed to delivery, since interpolation uses the private C registers and honest deletion certificates are simulated independently of data values. At an honest reconstruction, measuring the post-evaluation data directly gives the same outcomes as applying the prescribed evaluation to the pre-evaluation shares and then measuring them. The coherent

completion is a purification of the classical deferred sampling rule and retains every previously prepared source value. Thus tracing out the private purifying registers gives exactly the same delivered shares, reconstruction messages, and later classical disclosures in the two experiments. Hence both statistical distances are zero. □

Before proceeding, we take a moment to introduce the following notations. Define DEL_κ as the index set of the sharings in the κ-th batch that should be deleted in the reconstruction process. Concretely, $\mathsf{DEL}_\kappa = [M_\kappa - 2]$ for $\kappa \in [N^{\mathsf{mul}}]$, and $\mathsf{DEL}_\Gamma = [n+N]$. Moreover, for any subset $\mathcal{J} := \{j_1, \ldots, j_h\} \subseteq [m]$, where $1 \le j_1 < \cdots < j_h \le m$, define $\mathsf{C}^{(\kappa)}_{i,\iota_\kappa}[\mathcal{J}] := (\mathsf{C}^{(\kappa)}_{i,j_1,\iota_\kappa}, \ldots, \mathsf{C}^{(\kappa)}_{i,j_h,\iota_\kappa})$, and $\mathsf{A}^{(\kappa)}_{i,\iota_\kappa}[\mathcal{J}] := (\mathsf{A}^{(\kappa)}_{i,j_1,\iota_\kappa}, \ldots, \mathsf{A}^{(\kappa)}_{i,j_h,\iota_\kappa})$.

We retain all prepared private data registers until the end, including the never-delivered $\mathsf{A}^{(\kappa)}_{i,\iota}[\mathcal{J}_i^{L(\kappa)}]$ registers of honestly deleted parties. These private data registers are kept in post-evaluation coordinates, as specified in the implementation of Hyb_6 and Sim_6. Let $\mathcal{B} := \{i \in \mathcal{CS} : \varphi(i) = 1\}$ denote the set of parties whose quantum shares have been delivered to $\mathcal{A}$.

When an undeleted party $\mathcal{P}_i$, with $i = i_\upsilon^{(C)}$, is corrupted in $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$, for each $\kappa \in [\mathsf{cur}, \Gamma]$ and $\mathbf{x} \in \mathbb{Z}_q^{|\mathcal{K}^{(\kappa)}|}$, we write the coherent data-generation map before applying $\mathbf{V}_\kappa^{-1}$ as

$$|\mathbf{x}\rangle_{\mathsf{C}^{\mathsf{base},(\kappa)}_\iota} |\mathbf{0}\rangle_{\mathsf{A}^{(\kappa)}_{i,\iota}[\mathcal{J}_i^{\checkmark(\kappa)}]} \longmapsto |\mathbf{x}\rangle_{\mathsf{C}^{\mathsf{base},(\kappa)}_\iota} |\mathbf{L}_i^{(\kappa)}\mathbf{x} + \widetilde{f}_\iota^{(\kappa)}(0)\mathbf{l}_i^{(\kappa)}\rangle_{\mathsf{A}^{(\kappa)}_{i,\iota}[\mathcal{J}_i^{\checkmark(\kappa)}]},$$

where $\mathbf{L}_i^{(\kappa)}$ and $\mathbf{l}_i^{(\kappa)}$ are determined by the generation rule: if $\upsilon \le \hat{\tau}$, each row of $\mathbf{L}_i^{(\kappa)}$ selects the corresponding base coordinate and $\mathbf{l}_i^{(\kappa)} = \mathbf{0}$; otherwise, the coordinates on $\mathcal{J}_i^{L(\kappa)}$ are copied in the same way, while those on $\mathcal{J}_i^{R(\kappa)}$ are interpolated from $\{0\} \cup \mathcal{K}^{(\kappa)}$, with the corresponding rows of $\mathbf{L}_i^{(\kappa)}$ containing the Lagrange coefficients at the points in $\mathcal{K}^{(\kappa)}$ and the corresponding entries of $\mathbf{l}_i^{(\kappa)}$ containing the coefficient at 0. With a slight abuse of notation, let $\mathbf{L}_\mathcal{B}^{(\kappa)}$ and $\mathbf{l}_\mathcal{B}^{(\kappa)}$ denote the vertical concatenations of $\mathbf{L}_i^{(\kappa)}$ and $\mathbf{l}_i^{(\kappa)}$, respectively, over $i \in \mathcal{B}$ in a fixed party order. For each base label $\mathbf{x}$, let $|\mathbf{w}_\iota^{(\kappa)}(\mathbf{x})\rangle_{\mathsf{W}_\iota^{(\kappa)}}$ denote the computational-basis state of all remaining prepared private data registers in this coordinate, including any private register storing $\widetilde{f}_\iota^{(\kappa)}(0)$.

Put $\omega_q := e^{2\pi\sqrt{-1}/q}$. For $\mathbf{y} = (\mathbf{y}_i)_{i\in\mathcal{B}} \in (\mathbb{Z}_q^{m'})^\mathcal{B}$, define

$$|\mathsf{priv}_\iota^{(\kappa)}(\mathbf{y})\rangle := q^{-|\mathcal{K}^{(\kappa)}|/2} \sum_{\mathbf{x}\in\mathbb{Z}_q^{|\mathcal{K}^{(\kappa)}|}} \omega_q^{-\mathbf{y}^\mathsf{T}(\mathbf{L}_\mathcal{B}^{(\kappa)}\mathbf{x} + \widetilde{f}_\iota^{(\kappa)}(0)\mathbf{l}_\mathcal{B}^{(\kappa)})} |\mathbf{x}\rangle_{\mathsf{C}^{\mathsf{base},(\kappa)}_\iota} |\mathbf{w}_\iota^{(\kappa)}(\mathbf{x})\rangle_{\mathsf{W}_\iota^{(\kappa)}}. \tag{28}$$

The vectors $\mathbf{y}$ are Fourier-expansion indices; defining this family performs no measurement and introduces no new physical registers.

For any $\mathcal{D} \subseteq \mathcal{B}$ with $|\mathcal{B}\setminus\mathcal{D}| \le t$, write $\mathsf{cert}_\mathcal{D}^{(\kappa)} := \{\mathsf{cert}_{i,\iota}^{(\kappa)}\}_{i\in\mathcal{D},\,\iota\in[M_\kappa]}$ for the corresponding family of certificate vectors, and define

$$\mathcal{G}^{(\kappa)}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D}^{(\kappa)}) := \big\{\mathbf{y} \in (\mathbb{Z}_q^{m'})^\mathcal{B} \mid \mathsf{HW}(\mathbf{y}_i - \mathsf{cert}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{\checkmark(\kappa)}]) < m'_L/2 \text{ for all } i \in \mathcal{D}\big\}.$$

Define

$$\mathbf{\Pi}^{\mathsf{glob},(\kappa)}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D}^{(\kappa)}) := \sum_{\mathbf{y}\in\mathcal{G}^{(\kappa)}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}_\mathcal{D}^{(\kappa)})} |\mathsf{priv}_\iota^{(\kappa)}(\mathbf{y})\rangle\langle\mathsf{priv}_\iota^{(\kappa)}(\mathbf{y})|,$$

and set

$$\mathbf{\Pi}^{\mathsf{glob},(\kappa)}_{\mathcal{B},\mathcal{D}}(\mathsf{cert}^{(\kappa)}_{\mathcal{D}}) := \bigotimes_{\iota=1}^{M_\kappa} \mathbf{\Pi}^{\mathsf{glob},(\kappa)}_{\mathcal{B},\mathcal{D},\iota}(\mathsf{cert}^{(\kappa)}_{\mathcal{D}}). \tag{29}$$

We now define $\mathsf{Hyb}_{7,\tau'}(F,\mathbf{x})$ (resp. $\mathsf{Sim}_{7,\tau'}(F,\mathbf{x})$) and $\mathsf{Hyb}_{8,\tau'}(F,\mathbf{x})$ (resp. $\mathsf{Sim}_{8,\tau'}(F,\mathbf{x})$) for $\tau' \in [1,n]$. Define $\mathsf{Hyb}_{7,0}(F,\mathbf{x}) := \mathsf{Hyb}_6(F,\mathbf{x})$ and $\mathsf{Sim}_{7,0}(F,\mathbf{x}) := \mathsf{Sim}_6(F,\mathbf{x})$. In each experiment, initialize $\mathcal{D} := \emptyset$ to record the parties whose certificates have passed the additional measurements. All interpolated R coordinates delivered at corruption are computed from $\{0\} \cup \mathcal{K}^{(\kappa)}$.

- $\mathsf{Hyb}_{7,\tau'}(F,\mathbf{x})$ (resp. $\mathsf{Sim}_{7,\tau'}(F,\mathbf{x})$) for $\tau' \in [1,n]$:

  1. Suppose that, in $\mathsf{CD\text{-}phase}_\nu$, $\mathcal{P}_{i^{(D)}_{\tau'}}$ deletes its share and produces the deletion certificate $\mathsf{del_cert}_{i^{(D)}_{\tau'}} := \{\mathsf{cert}^{(\kappa)}_{i^{(D)}_{\tau'},\iota_\kappa}\}_{\kappa\in[\nu,\Gamma],\,\iota_\kappa\in[M_\kappa]}$. If all calls to Verify accept and $|\mathcal{CS}| < n$, let $\mathcal{D}^+ := \mathcal{D} \cup \{i^{(D)}_{\tau'}\}$. For every $\kappa \in [\nu,\Gamma]$ and every $j \in [m]$, let

  $$\mathbf{cert}^{(\kappa)}_{i^{(D)}_{\tau'}}[j] := (\mathsf{cert}^{(\kappa)}_{i^{(D)}_{\tau'},1}[j], \ldots, \mathsf{cert}^{(\kappa)}_{i^{(D)}_{\tau'},M_\kappa}[j])^T,$$

  and define

  $$\widetilde{\mathbf{cert}}^{(\kappa)}_{i^{(D)}_{\tau'}}[j] := \begin{cases} \mathbf{V}_\kappa^{-T}\mathbf{cert}^{(\kappa)}_{i^{(D)}_{\tau'}}[j], & j \in \mathcal{J}^{\checkmark(\kappa)}_{i^{(D)}_{\tau'}}, \\ \mathbf{cert}^{(\kappa)}_{i^{(D)}_{\tau'}}[j], & j \in \mathcal{J}^{\times(\kappa)}_{i^{(D)}_{\tau'}}. \end{cases}$$

  Let $\widetilde{\mathsf{cert}}^{(\kappa)}_{i^{(D)}_{\tau'},\iota_\kappa}$ be the ι_κ-th coordinate of these vectors over all $j \in [m]$.[6]

  Let $\widetilde{\mathsf{cert}}^{(\kappa)}_{\mathcal{D}^+} := \{\widetilde{\mathsf{cert}}^{(\kappa)}_{i,\iota}\}_{i\in\mathcal{D}^+,\,\iota\in[M_\kappa]}$, retaining the transformed certificates from earlier additional checks for the parties already in $\mathcal{D}$. The challenger $\mathcal{C}$ immediately performs the two-outcome measurement

  $$\left\{ \bigotimes_{\kappa=\nu}^{\Gamma} \mathbf{\Pi}^{\mathsf{glob},(\kappa)}_{\mathcal{B},\mathcal{D}^+}(\widetilde{\mathsf{cert}}^{(\kappa)}_{\mathcal{D}^+}),\ \mathbf{I} - \bigotimes_{\kappa=\nu}^{\Gamma} \mathbf{\Pi}^{\mathsf{glob},(\kappa)}_{\mathcal{B},\mathcal{D}^+}(\widetilde{\mathsf{cert}}^{(\kappa)}_{\mathcal{D}^+}) \right\}$$

  on the corresponding private data registers for every $\kappa \in [\nu,\Gamma]$ and $\iota \in [M_\kappa]$. If the result is "reject", then the challenger outputs $\perp$ and aborts; otherwise, it updates $\mathcal{D} \leftarrow \mathcal{D}^+$. If this deletion does not occur or $|\mathcal{CS}| = n$, no additional measurement is performed. No additional measurement acts on a batch after its scheduled reconstruction, or after successful finalization.

  2. All other steps follow $\mathsf{Hyb}_{7,\tau'-1}(F,\mathbf{x})$ (resp. $\mathsf{Sim}_{7,\tau'-1}(F,\mathbf{x})$).

For any $i \in [n], \kappa \in [\Gamma], \iota_\kappa \in [M_\kappa]$ and certificate vector $\mathsf{cert}^{(\kappa)}_{i,\iota_\kappa} \in \mathbb{Z}_q^m$, define

$$\mathbf{\Pi}^{\mathsf{loc},(\kappa)}(\mathsf{cert}^{(\kappa)}_{i,\iota_\kappa}) := \sum_{\mathbf{u}\in\mathbb{Z}_q^{m'}:\mathsf{HW}(\mathbf{u})<m'_L/2} (\mathsf{QFT}^{-1})^{\otimes m'}|\mathbf{u} + \mathsf{cert}^{(\kappa)}_{i,\iota_\kappa}[\mathcal{J}_i^{\checkmark(\kappa)}]\rangle\langle\mathbf{u} + \mathsf{cert}^{(\kappa)}_{i,\iota_\kappa}[\mathcal{J}_i^{\checkmark(\kappa)}]|\mathsf{QFT}^{\otimes m'}.$$

[6] If $\mathbf{V}_\kappa$ denotes the computational-basis transformation on the polynomial-evaluation registers, then $\mathbf{V}_\kappa\mathsf{QFT}^{\otimes M_\kappa}|\mathbf{a}\rangle = \mathsf{QFT}^{\otimes M_\kappa}|\mathbf{V}_\kappa^{-T}\mathbf{a}\rangle$. Hence, on the evaluation points, the Fourier-basis labels corresponding to the post-evaluation C-side registers are obtained by applying $\mathbf{V}_\kappa^{-T}$ to the deletion-certificate labels. The checking-point registers remain in the pre-evaluation coordinates, so their certificate labels are left unchanged. The affine offset is confined to the classical mask coordinates.

With a slight abuse of notation, for $\mathsf{cert}_i^{(\kappa)} := \{\mathsf{cert}_{i,\iota_\kappa}^{(\kappa)}\}_{\iota_\kappa \in [M_\kappa]}$, we define

$$\mathbf{\Pi}^{\mathsf{loc},(\kappa)}(\mathsf{cert}_i^{(\kappa)}) := \bigotimes_{\iota \in \mathsf{DEL}_\kappa} \mathbf{\Pi}^{\mathsf{loc},(\kappa)}(\mathsf{cert}_{i,\iota}^{(\kappa)}).$$

- $\mathsf{Hyb}_{8,\tau'}(F,\mathbf{x})$ (resp. $\mathsf{Sim}_{8,\tau'}(F,\mathbf{x})$) for $\tau' \in [1,n]$: This experiment is identical to $\mathsf{Hyb}_{7,\tau'}(F,\mathbf{x})$ (resp. $\mathsf{Sim}_{7,\tau'}(F,\mathbf{x})$), except the following. Whenever $\mathcal{A}$ corrupts an undeleted party $\mathcal{P}_i$ in $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$, instead of generating its R data by polynomial interpolation, the challenger $\mathcal{C}$ prepares, for every $\kappa \in [\mathsf{cur}, \Gamma]$, every $\iota_\kappa \in \mathsf{DEL}_\kappa$ and every $j \in \mathcal{J}_i^{R(\kappa)}$, the state

$$q^{-1/2} \sum_{x \in \mathbb{Z}_q} |x\rangle_{\mathsf{C}_{i,j,\iota_\kappa}^{(\kappa)}} \otimes |x\rangle_{\mathsf{A}_{i,j,\iota_\kappa}^{(\kappa)}}.$$

  It then applies $\mathbf{V}_\kappa^{-1}$ to each complete outgoing data block $(\mathsf{A}_{i,j,1}^{(\kappa)}, \ldots, \mathsf{A}_{i,j,M_\kappa}^{(\kappa)})$, including both L and R positions. It omits $\widetilde{f}_{\iota_\kappa}^{(\kappa)}(0)$ and any private register storing it for $\iota_\kappa \in \mathsf{DEL}_\kappa$. The coordinates in $[M_\kappa] \setminus \mathsf{DEL}_\kappa$ retain their polynomial generation. At each scheduled additional check for party i in $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$, put $\mathcal{D}^+ := \mathcal{D} \cup \{i\}$ and write $\widetilde{\mathsf{cert}}_i := (\widetilde{\mathsf{cert}}_i^{(\kappa)})_{\kappa=\mathsf{cur}}^{\Gamma}$ and $\widetilde{\mathsf{cert}}_{\mathcal{D}^+} := (\widetilde{\mathsf{cert}}_{\mathcal{D}^+}^{(\kappa)})_{\kappa=\mathsf{cur}}^{\Gamma}$. Define the projectors on the coordinates to be deleted and reconstructed, respectively, by

$$\mathbf{\Pi}^{\mathsf{del}}(\widetilde{\mathsf{cert}}_i) := \bigotimes_{\kappa=\mathsf{cur}}^{\Gamma} \bigotimes_{\iota \in \mathsf{DEL}_\kappa} \mathbf{\Pi}^{\mathsf{loc},(\kappa)}(\widetilde{\mathsf{cert}}_{i,\iota}^{(\kappa)}),$$

$$\mathbf{\Pi}^{\mathsf{rec}}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+}) := \bigotimes_{\kappa=\mathsf{cur}}^{\Gamma} \bigotimes_{\iota \in [M_\kappa] \setminus \mathsf{DEL}_\kappa} \mathbf{\Pi}_{\mathcal{B},\mathcal{D}^+,\iota}^{\mathsf{glob},(\kappa)}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+}^{(\kappa)}).$$

  Replace the additional measurement of $\mathsf{Hyb}_{7,\tau'}$ (resp. $\mathsf{Sim}_{7,\tau'}$) by the two-outcome measurement with acceptance projector

$$\mathbf{\Pi}^{\mathsf{del}}(\widetilde{\mathsf{cert}}_i) \otimes \mathbf{\Pi}^{\mathsf{rec}}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+}),$$

  with the same rejection and update rules. Honest deletion certificates continue to be simulated as in Hyb_4 (resp. Sim_4).

Let $\mathcal{COR}(\upsilon)$ denote the processing of the corruption query for $\mathcal{P}_{i_\upsilon^{(C)}}$, and let $\mathcal{DEL}(\upsilon')$ denote the processing of the corrupt-then-delete query for $\mathcal{P}_{i_{\upsilon'}^{(D)}}$, including any scheduled additional check. These counters refer only to computing-party events before successful finalization. Honest deletions do not advance either counter; corruptions after honest deletion advance only the corruption counter. Every view includes the transcript and abort outcome.

For every fixed $\tau' \in [1,n]$ and $k \in \{7,8\}$, define $\overline{\mathsf{Hyb}}_{k,\tau'}(F,\mathbf{x})$ and $\overline{\mathsf{Sim}}_{k,\tau'}(F,\mathbf{x})$ to follow $\mathsf{Hyb}_{k,\tau'}(F,\mathbf{x})$ and $\mathsf{Sim}_{k,\tau'}(F,\mathbf{x})$, respectively, except that they stop immediately before processing the first corruption query $\mathcal{COR}(\tau)$ with $\eta(\tau) > \tau'$ during $\mathsf{CD\text{-}phase}_\nu$ for $\nu \in [\Gamma]$. Upon stopping, they output the adversary's current view.[7] If this stopping condition is never triggered, they return the complete terminating view, including any abort and all post-finalization corruptions.

[7] Unlike the truncated secret-sharing experiments, the truncated MPC experiments need no cutoff after all parties are corrupted: online honest majority excludes this case before finalization, and post-finalization corruptions reveal only residual classical states.

We first prove the following analogue of Lemma 32.

Lemma 45. *For every fixed* $\tau' \in [1,n]$, $\mathsf{TD}\big(\overline{\mathsf{Hyb}}_{7,\tau'}(F,\mathbf{x}), \overline{\mathsf{Hyb}}_{8,\tau'}(F,\mathbf{x})\big) = 0$, *and* $\mathsf{TD}\big(\overline{\mathsf{Sim}}_{7,\tau'}(F,\mathbf{x}), \overline{\mathsf{Sim}}_{8,\tau'}(F,\mathbf{x})\big) = 0$.

Proof. The proof is similar to that of Lemma 32. Fix one of the two pairs of experiments, and denote it by

$$(\mathsf{G}_{7,\tau'}, \mathsf{G}_{8,\tau'}) \in \left\{ \big(\mathsf{Hyb}_{7,\tau'}, \mathsf{Hyb}_{8,\tau'}\big), \big(\mathsf{Sim}_{7,\tau'}, \mathsf{Sim}_{8,\tau'}\big) \right\}.$$

Fix the common classical setup, the initial secret values, and the classical masks. We prove the claim for every such choice and then average over these values. We first describe the state of $\overline{\mathsf{G}}_{8,\tau'}$ and define its Fourier coefficient states. We then show that $\overline{\mathsf{G}}_{7,\tau'}$ has a corresponding expansion with exactly these coefficient states.

Let R collect the side registers, including the adversary's registers, the checking and classical registers, all registers in coordinates $[M_\kappa] \setminus \mathsf{DEL}_\kappa$ for every κ, and the remaining purifying registers, but excluding the private data registers in DEL_κ.

For each $\mathsf{cur} \in [\Gamma]$, our analysis traces out the private data registers of the cur-th batch in coordinates $\mathsf{DEL}_{\mathsf{cur}}$ at the end of $\mathsf{Round}_{\mathsf{cur}}$; these registers are never accessed or disclosed afterward. The states $|\Psi^{\mathsf{G}_8}_{\mathcal{B},\mathcal{D}}\rangle$ and $|\Psi^{\mathsf{G}_7}_{\mathcal{B},\mathcal{D}}\rangle$ below purify the remaining systems. Auxiliary registers introduced for this purification are included in R and are never accessed afterward.

We first focus on $\overline{\mathsf{G}}_{8,\tau'}$. Immediately after a corruption or deletion event in $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$, let $\mathcal{B}$ and $\mathcal{D}$ denote the current sets. We suppress the independent private EPR pairs corresponding to data registers in DEL_κ that have not been delivered. The current state admits an unnormalized purification

$$|\Psi^{\mathsf{G}_8}_{\mathcal{B},\mathcal{D}}\rangle = \sum_{\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota}} \left(\bigotimes_{\kappa=\mathsf{cur}}^{\Gamma} \bigotimes_{\iota\in\mathsf{DEL}_\kappa} \bigotimes_{i\in\mathcal{B}} (\mathsf{QFT}^{-1})^{\otimes m'} |\mathbf{y}_{i,\iota}^{(\kappa)}\rangle_{\mathsf{C}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{\checkmark(\kappa)}]} \right) |\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota})\rangle_{\mathsf{R}}, \tag{30}$$

where the sum ranges over $\mathbf{y}_\iota^{(\kappa)} \in \mathcal{G}^{(\kappa)}_{\mathcal{B},\mathcal{D},\iota}(\widetilde{\mathsf{cert}}^{(\kappa)}_{\mathcal{D}})$ for every $\kappa \in [\mathsf{cur},\Gamma]$ and $\iota \in \mathsf{DEL}_\kappa$. For each allowed label family, this expansion defines its coefficient state, which may be unnormalized.

Initially, $\mathcal{B} = \mathcal{D} = \emptyset$. There is only the empty label, with coefficient state $|\chi_{\emptyset,\emptyset}(\emptyset)\rangle_{\mathsf{R}}$ containing the initial side information, so Eqn (30) holds.

For the induction step, each operation on R is represented on each outcome branch by a Kraus operator $\mathbf{K}$, after including any necessary purifying registers in R. We now consider the corruption and deletion events.

- At $\mathcal{COR}(\upsilon)$ in $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$, write $i := i_\upsilon^{(C)}$.

  If $\varphi(i) = 0$, only classical information is disclosed, so the preceding observation applies and $\mathcal{B},\mathcal{D}$ remain unchanged. Otherwise, for each $\kappa \in [\mathsf{cur},\Gamma]$ and $\iota \in \mathsf{DEL}_\kappa$, the data EPR pairs

prepared for this delivery satisfy

$$q^{-m'/2}\sum_{\mathbf{x}\in\mathbb{Z}_q^{m'}}|\mathbf{x}\rangle_{\mathsf{C}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{\checkmark(\kappa)}]}|\mathbf{x}\rangle_{\mathsf{A}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{\checkmark(\kappa)}]}$$
$$=q^{-m'/2}\sum_{\mathbf{y}_\iota^{(\kappa)+}\in\mathbb{Z}_q^{m'}}\big((\mathsf{QFT}^{-1})^{\otimes m'}|\mathbf{y}_\iota^{(\kappa)+}\rangle_{\mathsf{C}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{\checkmark(\kappa)}]}\big)\otimes\big(\mathsf{QFT}^{\otimes m'}|\mathbf{y}_\iota^{(\kappa)+}\rangle_{\mathsf{A}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{\checkmark(\kappa)}]}\big).$$

Put $\mathcal{B}^+:=\mathcal{B}\cup\{i\}$. Let $\mathbf{K}$ represent the side operations on the current outcome branch, including the generation of the outgoing coordinates in $[M_\kappa]\setminus\mathsf{DEL}_\kappa$. Let

$$\widehat{\mathbf{V}}_{i,\mathsf{cur}}:=\bigotimes_{\kappa=\mathsf{cur}}^{\Gamma}\bigotimes_{j\in\mathcal{J}_i^{\checkmark(\kappa)}}\mathbf{V}_\kappa^{-1},$$

where each factor acts on the complete outgoing data block $(\mathsf{A}_{i,j,1}^{(\kappa)},\dots,\mathsf{A}_{i,j,M_\kappa}^{(\kappa)})$. After including the new A registers in R, define the new coefficient states by

$$|\chi_{\mathcal{B}^+,\mathcal{D}}(\{(\mathbf{y}_\iota^{(\kappa)},\mathbf{y}_\iota^{(\kappa)+})\}_{\kappa,\iota})\rangle:=$$
$$q^{-\frac{m'}{2}\sum_{\kappa=\mathsf{cur}}^{\Gamma}|\mathsf{DEL}_\kappa|}\widehat{\mathbf{V}}_{i,\mathsf{cur}}\Bigg(\mathbf{K}|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota})\rangle_{\mathsf{R}}\otimes\bigotimes_{\substack{\kappa\in[\mathsf{cur},\Gamma]\\ \iota\in\mathsf{DEL}_\kappa}}\mathsf{QFT}^{\otimes m'}|\mathbf{y}_\iota^{(\kappa)+}\rangle_{\mathsf{A}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{\checkmark(\kappa)}]}\Bigg).\tag{31}$$

Since $i\notin\mathcal{D}$, we have $(\mathbf{y}_\iota^{(\kappa)},\mathbf{y}_\iota^{(\kappa)+})\in\mathcal{G}_{\mathcal{B}^+,\mathcal{D},\iota}^{(\kappa)}(\widetilde{\mathsf{cert}}_{\mathcal{D}}^{(\kappa)})$. This gives precisely the required expansion after corruption.

- At $\mathcal{DEL}(\upsilon')$ in $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$, write $i:=i_{\upsilon'}^{(D)}$. Let $\mathbf{K}$ represent the side operations on the current outcome branch, including certificate generation and verification using Verify.

  If any call to Verify rejects, the experiment aborts. If no additional check is performed, the coefficient states are simply updated by $\mathbf{K}$.

  Otherwise, put $\mathcal{D}^+:=\mathcal{D}\cup\{i\}$. The accepting state is

$$|\Psi_{\mathcal{B},\mathcal{D}^+}^{\mathsf{G}_8}\rangle=\big(\mathbf{\Pi}^{\mathsf{del}}(\widetilde{\mathsf{cert}}_i)\otimes\mathbf{\Pi}^{\mathsf{rec}}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+})\big)(\mathbf{I}\otimes\mathbf{K})|\Psi_{\mathcal{B},\mathcal{D}}^{\mathsf{G}_8}\rangle.$$

  Here $\mathbf{\Pi}^{\mathsf{del}}(\widetilde{\mathsf{cert}}_i)$ keeps exactly the terms with $\mathbf{y}_\iota^{(\kappa)}\in\mathcal{G}_{\mathcal{B},\mathcal{D}^+,\iota}^{(\kappa)}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+}^{(\kappa)})$ for every $\kappa\in[\mathsf{cur},\Gamma]$ and $\iota\in\mathsf{DEL}_\kappa$, while $\mathbf{\Pi}^{\mathsf{rec}}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+})$ acts on R.

  Thus the accepting coefficient states are

$$|\chi_{\mathcal{B},\mathcal{D}^+}(\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota})\rangle:=\mathbf{\Pi}^{\mathsf{rec}}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+})\mathbf{K}|\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota})\rangle$$

  on the new allowed set.

Honest deletions and scheduled reconstruction act on R and preserve the expansion.

At the end of $\mathsf{Round}_{\mathsf{cur}}$, we trace out the private data registers of the cur-th batch in coordinates $\mathsf{DEL}_{\mathsf{cur}}$ and purify the remaining joint state as above. This gives the expansion for the next phase, with an empty label family after the final round.

We now consider $\overline{\mathsf{G}}_{7,\tau'}$. Use the same purifications of the side operations as in $\overline{\mathsf{G}}_{8,\tau'}$. We prove by induction over the corruption and deletion events and the scheduled protocol operations that every sequence of events and outcomes occurs with the same probability in the two experiments, including the outcomes of the additional checks.

Conditioned on no abort having occurred, the state in $\overline{\mathsf{G}}_{7,\tau'}$ during $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$ has a purification of the form

$$|\Psi^{\mathsf{G}_7}_{\mathcal{B},\mathcal{D}}\rangle = \sum_{\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota}} \left(\bigotimes_{\kappa=\mathsf{cur}}^{\Gamma} \bigotimes_{\iota\in\mathsf{DEL}_\kappa} |\mathsf{priv}_\iota^{(\kappa)}(\mathbf{y}_\iota^{(\kappa)})\rangle \right) |\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota})\rangle_{\mathsf{R}}, \tag{32}$$

with exactly the coefficient states defined in $\overline{\mathsf{G}}_{8,\tau'}$ for the same events and outcomes. The label ranges are the same as in Eqn (30). As in the proof of Lemma 32, the family $\{|\mathsf{priv}_\iota^{(\kappa)}(\mathbf{y})\rangle\}_{\mathbf{y}\in\mathcal{G}^{(\kappa)}_{\mathcal{B},\mathcal{D},\iota}(\widetilde{\mathsf{cert}}^{(\kappa)}_{\mathcal{D}})}$ is orthonormal for every $\kappa \in [\mathsf{cur},\Gamma]$ and $\iota \in \mathsf{DEL}_\kappa$.

Initially, for each κ and $\iota \in \mathsf{DEL}_\kappa$, $|\mathsf{priv}_\iota^{(\kappa)}(\emptyset)\rangle$ contains the private L EPR pairs, together with any private register storing $\widetilde{f}_\iota^{(\kappa)}(0)$. The initial side state is exactly $|\chi_{\emptyset,\emptyset}(\emptyset)\rangle_{\mathsf{R}}$, so the claimed expansion holds.

- At $\mathcal{COR}(\upsilon)$ in $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$, write $i := i_\upsilon^{(C)}$.

  If $\varphi(i) = 0$, only classical information is disclosed, so the common side operations preserve the claimed expansion.

  Otherwise, let $\mathbf{K}$ denote the same side operations as in $\overline{\mathsf{G}}_{8,\tau'}$, acting on the old R up to the moment before delivery of the DEL_κ data registers. (Such a common $\mathbf{K}$ exists because, by the induction hypothesis, the adversary's views in the two experiments are identical up to this point, and the challenger's remaining side operations have the same specification.)

  For $\upsilon \le \hat{\tau}$, the desired identity follows from the same EPR expansion as in $\overline{\mathsf{G}}_{8,\tau'}$.

  We now consider $\upsilon > \hat{\tau}$, when the base registers remain unchanged. Fix $\kappa \in [\mathsf{cur},\Gamma]$ and $\iota \in \mathsf{DEL}_\kappa$, and let $\mathbf{Q}_i$ denote this coordinate's data-generation and delivery operation before applying $\mathbf{V}_\kappa^{-1}$. Write

  $$\mathsf{W}_\iota^{(\kappa)} = \mathsf{W}_\iota^{(\kappa)-} \otimes \mathsf{A}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{L(\kappa)}], \qquad \mathsf{W}_\iota^{(\kappa)+} := \mathsf{W}_\iota^{(\kappa)-} \otimes \mathsf{C}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{R(\kappa)}].$$

  Let $\mathbf{w}_\iota^{(\kappa)-}(\mathbf{x})$ be the restriction of $\mathbf{w}_\iota^{(\kappa)}(\mathbf{x})$ to the old registers in $\mathsf{W}_\iota^{(\kappa)-}$, and put

  $$|\mathbf{w}_\iota^{(\kappa)+}(\mathbf{x})\rangle := |\mathbf{w}_\iota^{(\kappa)-}(\mathbf{x})\rangle \otimes |\mathbf{L}_i^{R(\kappa)}\mathbf{x} + \widetilde{f}_\iota^{(\kappa)}(0)\mathbf{l}_i^{R(\kappa)}\rangle_{\mathsf{C}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{R(\kappa)}]},$$

  where $\mathbf{L}_i^{R(\kappa)}$ and $\mathbf{l}_i^{R(\kappa)}$ are the rows and entries corresponding to $\mathcal{J}_i^{R(\kappa)}$. Expanding the input private state, we obtain

  $$\begin{aligned}
  &\mathbf{Q}_i|\mathsf{priv}_\iota^{(\kappa)}(\mathbf{y})\rangle \\
  &\quad = q^{-|\mathcal{K}^{(\kappa)}|/2} \sum_{\mathbf{x}\in\mathbb{Z}_q^{|\mathcal{K}^{(\kappa)}|}} \omega_q^{-\mathbf{y}^{\mathsf{T}}(\mathbf{L}_{\mathcal{B}}^{(\kappa)}\mathbf{x}+\widetilde{f}_\iota^{(\kappa)}(0)\mathbf{l}_{\mathcal{B}}^{(\kappa)})} |\mathbf{x}\rangle_{\mathsf{C}_\iota^{\mathsf{base},(\kappa)}} |\mathbf{w}_\iota^{(\kappa)+}(\mathbf{x})\rangle \\
  &\qquad \otimes |\mathbf{L}_i^{(\kappa)}\mathbf{x} + \widetilde{f}_\iota^{(\kappa)}(0)\mathbf{l}_i^{(\kappa)}\rangle_{\mathsf{A}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{\checkmark(\kappa)}]} \\
  &\quad = q^{-m'/2} \sum_{\mathbf{y}_\iota^{(\kappa)+}\in\mathbb{Z}_q^{m'}} |\mathsf{priv}_\iota^{(\kappa)}(\mathbf{y},\mathbf{y}_\iota^{(\kappa)+})\rangle \otimes \mathsf{QFT}^{\otimes m'} |\mathbf{y}_\iota^{(\kappa)+}\rangle_{\mathsf{A}_{i,\iota}^{(\kappa)}[\mathcal{J}_i^{\checkmark(\kappa)}]}.
  \end{aligned} \tag{33}$$

The private vectors in the last line use the updated registers $\mathsf{C}_\iota^{\mathsf{base},(\kappa)} \otimes \mathsf{W}_\iota^{(\kappa)+}$.

The last equality follows by expanding the outgoing data in the Fourier basis: its phase combines with the old phase exactly as in Eqn (16).

Applying this identity to every $\kappa \in [\mathsf{cur}, \Gamma]$ and $\iota \in \mathsf{DEL}_\kappa$, and then applying the same $\widehat{\mathbf{V}}_{i,\mathsf{cur}}$, gives exactly the coefficient states in Eqn (31).

This includes the mixing of the replaced and unreplaced coordinates by $\mathbf{V}_\kappa^{-1}$ and proves the claimed correspondence after corruption.

- At $\mathcal{DEL}(\upsilon')$ in $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$, write $i := i_{\upsilon'}^{(D)}$. Let $\mathbf{K}$ denote the same side operations as in $\overline{\mathsf{G}}_{8,\tau'}$, including certificate generation and verification using Verify.

  If any call to Verify rejects, both experiments abort. If no additional check is performed, the coefficient states are updated by the same $\mathbf{K}$.

  Otherwise, put $\mathcal{D}^+ := \mathcal{D} \cup \{i\}$.

  The private families are unchanged at this check and are orthonormal on the old allowed sets. Thus the global projectors on DEL_κ keep exactly the labels in the new allowed sets, as does $\mathbf{\Pi}^{\mathsf{del}}(\widetilde{\mathsf{cert}}_i)$ in $\overline{\mathsf{G}}_{8,\tau'}$.

  The remaining factors form the same $\mathbf{\Pi}^{\mathsf{rec}}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+})$ on R. The accepting joint state is

$$\sum_{\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota}} \left(\bigotimes_{\kappa=\mathsf{cur}}^{\Gamma} \bigotimes_{\iota \in \mathsf{DEL}_\kappa} |\mathsf{priv}_\iota^{(\kappa)}(\mathbf{y}_\iota^{(\kappa)})\rangle \right) \otimes \mathbf{\Pi}^{\mathsf{rec}}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+}) \mathbf{K} |\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota})\rangle_{\mathsf{R}}, \tag{34}$$

  where the sum ranges over $\mathbf{y}_\iota^{(\kappa)} \in \mathcal{G}^{(\kappa)}_{\mathcal{B},\mathcal{D}^+,\iota}(\widetilde{\mathsf{cert}}^{(\kappa)}_{\mathcal{D}^+})$ for every $\kappa \in [\mathsf{cur}, \Gamma]$ and $\iota \in \mathsf{DEL}_\kappa$. For each retained label family, the coefficient state is

$$|\chi_{\mathcal{B},\mathcal{D}^+}(\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota})\rangle_{\mathsf{R}} = \mathbf{\Pi}^{\mathsf{rec}}(\widetilde{\mathsf{cert}}_{\mathcal{D}^+}) \mathbf{K} |\chi_{\mathcal{B},\mathcal{D}}(\{\mathbf{y}_\iota^{(\kappa)}\}_{\kappa,\iota})\rangle_{\mathsf{R}},$$

  exactly as defined in $\overline{\mathsf{G}}_{8,\tau'}$. Thus acceptance gives the required state $|\Psi^{\mathsf{G}_7}_{\mathcal{B},\mathcal{D}^+}\rangle$.

  By orthonormality, acceptance gives the same unnormalized reduced state on R as in $\overline{\mathsf{G}}_{8,\tau'}$. Since rejection uses the complementary projectors, the same coefficient-state correspondence implies that the unnormalized reduced state on R after rejection is also identical to that in $\overline{\mathsf{G}}_{8,\tau'}$.

All remaining operations, including honest deletions, client corruptions, reconstruction, and post-finalization disclosures, act on the common side system R and preserve the coefficient-state correspondence.

The global projectors and coherent completion preserve support on degree-at-most-d polynomials with the prescribed constant terms, so reconstruction still returns the prescribed value or aborts.

At the end of $\mathsf{Round}_{\mathsf{cur}}$, we trace out the private data registers of the cur-th batch in coordinates $\mathsf{DEL}_{\mathsf{cur}}$. By orthonormality of the private families, the remaining joint state admits a purification with the same coefficient states as in $\overline{\mathsf{G}}_{8,\tau'}$. The correspondence therefore persists at the next phase and, after the final round, gives the same side state for all subsequent disclosures.

This completes the induction.

We next state the analogue of Lemma 33.

Lemma 46. *Put* $\varepsilon := 2\sum_{\kappa=1}^{\Gamma} M_\kappa \exp(-\ell(m'_L)^2/(2m^2))$. *For every* $\tau' \in [1,n]$,

$$\mathsf{TD}\big(\mathsf{Hyb}_{7,\tau'-1}(F,\mathbf{x}), \mathsf{Hyb}_{7,\tau'}(F,\mathbf{x})\big) \le 2\sqrt{\varepsilon} + \varepsilon = 2^{-\Omega(\lambda)}$$

and

$$\mathsf{TD}\big(\mathsf{Sim}_{7,\tau'-1}(F,\mathbf{x}), \mathsf{Sim}_{7,\tau'}(F,\mathbf{x})\big) \le 2\sqrt{\varepsilon} + \varepsilon = 2^{-\Omega(\lambda)}.$$

Proof. Fix $\mathsf{G} \in \{\mathsf{Hyb}, \mathsf{Sim}\}$ and the initial secret values, client-side values, and classical masks. This fixes all $\mathbf{V}_\kappa$ independently of the verification setup and nonconstant polynomial coefficients. The two experiments differ only in the additional check at $\mathcal{DEL}(\tau')$. Let p_{bad} be the probability that this deletion is reached with $|\mathcal{CS}| < n$, all calls to Verify accept, and the additional check rejects. As in Lemma 33, it suffices to prove $p_{\mathsf{bad}} \le \varepsilon$.

For every $\mathsf{cur} \in [\Gamma]$, define the following auxiliary experiment.

- $\widetilde{\mathsf{G}}^{(\mathsf{cur})}$: This experiment is identical to $\mathsf{G}_{7,\tau'}$, except that, for every $\kappa \in [\mathsf{cur}, \Gamma]$ and every $\iota \in [M_\kappa]$, the challenger replaces the data-generation procedure from the beginning by fresh EPR pairs. It delivers the A halves after applying $\mathbf{V}_\kappa^{-1}$ to each complete outgoing data block as before. For these batches, it generates the verification information lazily, as in the proof of Lemma 33.

  At every additional check, it replaces each corresponding global projector by the local projector $\mathbf{\Pi}^{\mathsf{loc},(\kappa)}(\widetilde{\mathsf{cert}}^{(\kappa)}_{i,\iota})$, where i is the party being deleted.

  The experiment stops immediately after processing $\mathcal{DEL}(\tau')$ or immediately before the scheduled processing of batch cur, whichever occurs first.

The difference between $\widetilde{\mathsf{G}}^{(\mathsf{cur})}$ and $\mathsf{G}_{8,\tau'}$ is not confined to the lazy generation at checking points. At evaluation points, for every $\kappa \in [\mathsf{cur}, \Gamma]$, $\widetilde{\mathsf{G}}^{(\mathsf{cur})}$ replaces all sharing coordinates in $[M_\kappa]$, including $[M_\kappa] \setminus \mathsf{DEL}_\kappa$, whereas $\mathsf{G}_{8,\tau'}$ replaces only those in DEL_κ.

No replaced batch is reconstructed before the stopping point, and legality together with the preceding checks gives $|\mathcal{B} \setminus \mathcal{D}| \le t$ before each additional check. The proof of Lemma 32 therefore applies jointly to these batches, with the other batches included in the side registers. It shows that replacing polynomial generation by fresh EPR pairs and the global projectors by local projectors preserves the adversary's view and the probabilities of measurement outcomes. Generating the verification information lazily also preserves these distributions, by the argument in the proof of Lemma 33. Thus verification using Verify and the additional check at $\mathcal{DEL}(\tau')$ have the same joint outcome probabilities in $\widetilde{\mathsf{G}}^{(\mathsf{cur})}$ and $\mathsf{G}_{7,\tau'}$ whenever this deletion occurs in $\mathsf{CD\text{-}phase}_{\mathsf{cur}}$.

The sampling calculation follows the proof of Lemma 33. Fix $\kappa \in [\mathsf{cur}, \Gamma]$ and $\iota \in [M_\kappa]$, and write $i := i^{(D)}_{\tau'}$. Conceptually measure the private data registers of coordinate ι directly in the inverse-QFT basis, obtaining the post-evaluation labels $y^{(\kappa)}_{i,\iota}$. For every $j \in \mathcal{J}_i^{\checkmark(\kappa)}$, define

$$\zeta_j := \mathbf{1}\Big\{ y^{(\kappa)}_{i,\iota}[j] \neq \widetilde{\mathsf{cert}}^{(\kappa)}_{i,\iota}[j] \Big\}.$$

For the sampling analysis, extend these indicators to the remaining hidden checking positions by applying $\mathbf{V}_\kappa^{-T}$ to both their inverse-Fourier outcome vectors and their certificate vectors, and

comparing coordinate ι. This classical relabeling gives the same EPR measurement description at data and checking positions. The lazy-sampling argument of Lemma 33 therefore applies to these indicators.

Rejection of the local projector for coordinate ι implies that at least $m'_L/2$ of these indicators equal one on data positions. If every call to Verify accepts, then $\zeta_j = 0$ on every remaining hidden checking position. Remark 35 therefore bounds this joint event by $2\exp(-\ell(m'_L)^2/(2m^2))$. A union bound over batches and sharing coordinates yields

$$p_{\mathsf{bad}} \le 2\sum_{\kappa=1}^{\Gamma} M_\kappa \exp\left(-\frac{\ell(m'_L)^2}{2m^2}\right) = \varepsilon.$$

□

Lemma 47. $\mathsf{TD}\big(\mathsf{Hyb}_{8,n}(F,\mathbf{x}), \mathsf{Sim}_{8,n}(F,\mathbf{x})\big) = 0.$

Proof. Fix the clients' inputs and their public masked input messages. In $\mathsf{Hyb}_{8,n}$, the multiplication reconstructions are $(u'_\kappa, v'_\kappa) = (u_\kappa + s_1, v_\kappa + s_2)$, where s_1, s_2 depend only on the inputs and preceding masks, and u_κ, v_κ are fresh independent uniform masks. The output reconstructions are $o'_h = o_h + \gamma_h$, with fresh independent uniform γ_h. Hence the prescribed reconstructed values are jointly uniform. The same holds in $\mathsf{Sim}_{8,n}$: each batch has a fresh uniform secret vector, and its invertible affine evaluation map is determined by preceding prescribed reconstructions.

By the definitions of $\mathsf{Hyb}_{8,n}$ and $\mathsf{Sim}_{8,n}$, the remaining view is generated in the same way from these prescribed reconstructed values, the independent uniform masks $\{\sigma^{(\kappa)}\}_{\kappa\in[\Gamma]}$, and the fixed client-side information. Hence the complete adversarial views are identically distributed. □

Finally, we prove Lemma 40 as follows.

Proof of Lemma 40.

$$\begin{aligned}
&\mathsf{TD}(\mathsf{Hyb}_2(F,\mathbf{x}), \mathsf{Sim}_2(F,\mathbf{x}))\\
=&\mathsf{TD}(\mathsf{Hyb}_3(F,\mathbf{x}), \mathsf{Sim}_3(F,\mathbf{x})) \quad \text{(Lemma 41)}\\
=&\mathsf{TD}(\mathsf{Hyb}_6(F,\mathbf{x}), \mathsf{Sim}_6(F,\mathbf{x})) \quad \text{(Lemmas 42, 43, and 44)}\\
=&\mathsf{TD}(\mathsf{Hyb}_{7,0}(F,\mathbf{x}), \mathsf{Sim}_{7,0}(F,\mathbf{x}))\\
\le&\mathsf{TD}(\mathsf{Hyb}_{7,n}(F,\mathbf{x}), \mathsf{Sim}_{7,n}(F,\mathbf{x})) + 2^{-\Omega(\lambda)} \quad \text{(Lemma 46)}\\
\le&\mathsf{TD}(\mathsf{Hyb}_{7,n}(F,\mathbf{x}), \mathsf{Hyb}_{8,n}(F,\mathbf{x})) + \mathsf{TD}(\mathsf{Hyb}_{8,n}(F,\mathbf{x}), \mathsf{Sim}_{8,n}(F,\mathbf{x}))\\
&\quad + \mathsf{TD}(\mathsf{Sim}_{8,n}(F,\mathbf{x}), \mathsf{Sim}_{7,n}(F,\mathbf{x})) + 2^{-\Omega(\lambda)} \quad \text{(triangle inequality)}\\
=&\mathsf{TD}(\mathsf{Hyb}_{8,n}(F,\mathbf{x}), \mathsf{Sim}_{8,n}(F,\mathbf{x})) + 2^{-\Omega(\lambda)} \quad \text{(Lemma 45)}\\
=&2^{-\Omega(\lambda)} \quad \text{(Lemma 47).}
\end{aligned}$$

□

5.5 Multiplication-Depth-Based Batching

We present a more efficient variant of the protocol in Section 5.2, using a technique that we call *multiplication-depth-based batching*. The main idea is to reconstruct, from a single secret-sharing

batch, the masked values associated with all multiplication gates of the same multiplication depth. This reduces the number of batches from $N^{\mathsf{mul}} + 1$ to $D + 1$, where D denotes the multiplication depth of the circuit.

For each $\iota \in [n + N]$, let depth_ι denote the multiplication depth of WIRE_ι, defined recursively as follows:

- If $\iota \in [n]$, then $\mathsf{depth}_\iota := 0$.
- If WIRE_ι is the output wire of a linear gate with input wires WIRE_{ι_1} and WIRE_{ι_2}, where $\iota_1 = \iota_2$ for a unary gate, then $\mathsf{depth}_\iota := \max\{\mathsf{depth}_{\iota_1}, \mathsf{depth}_{\iota_2}\}$.
- If WIRE_ι is the output wire of a multiplication gate with input wires WIRE_{ι_1} and WIRE_{ι_2}, then $\mathsf{depth}_\iota := \max\{\mathsf{depth}_{\iota_1}, \mathsf{depth}_{\iota_2}\} + 1$.

For each gate GATE_g, let D_g be the multiplication depth of its output wire. Define $D := \max_{g\in[N]} D_g$ and $\Gamma := D + 1$. We partition the gates into Γ layers: for each $\kappa \in [\Gamma]$, the κ-th layer contains all linear gates whose output wires have multiplication depth $\kappa - 1$, together with, when $\kappa \in [D]$, all multiplication gates whose output wires have multiplication depth κ.

We relabel the gates by ordering them first by layer and then topologically within each layer; this is still a topological ordering. We relabel their output wires accordingly, so that the output wire of GATE_g remains WIRE_{n+g}. For every $\kappa \in [\Gamma]$, let N_κ be the number of gates contained in the first κ layers, with $N_0 := 0$ and $N_\Gamma = N$. For every $\kappa \in [D]$, denote the multiplication gates in the κ-th layer by

$$\mathsf{GATE}_{l_{\kappa,1}}, \ldots, \mathsf{GATE}_{l_{\kappa,N_\kappa^{\mathsf{mul}}}}, \qquad N_{\kappa-1} < l_{\kappa,1} < \cdots < l_{\kappa,N_\kappa^{\mathsf{mul}}} \le N_\kappa.$$

Let n' be the number of circuit outputs. For every $\kappa \in [\Gamma]$, define

$$M_\kappa := \begin{cases} n + N_\kappa + N_\kappa^{\mathsf{mul}}, & \kappa \in [D], \\ n + N + n', & \kappa = \Gamma. \end{cases}$$

Thus, for every $\kappa \in [D]$, the κ-th batch contains one coordinate for each input wire, one coordinate for each gate in the first κ layers, and one additional coordinate for each multiplication gate in the κ-th layer. The final batch contains one coordinate for each input and gate wire, together with n' additional coordinates for the output masks.

For every $\kappa \in [\Gamma]$, let $\Pi^{(\kappa)} := \Pi^{\mathsf{SSB-UVD}}_{n,t,M_\kappa,\mathbb{Z}_q}$ denote an $(n, t, M_\kappa, \mathbb{Z}_q)$-secret-sharing batch scheme defined in Section 4.3. For the classical output-mask sharings, the protocol invokes the adaptively programmable classical secret-sharing protocol with robust opening Π_{ROpenSS} from Definition 14. All other notation and the convention on public classical masks are inherited from Section 5.2.

We construct the protocol $\widetilde{\Pi}^{\mathsf{MPC}}_F$ as follows. As an illustrative example, we depict in Figure 3 the circuit corresponding to $F(x_1, x_2, x_3, x_4, x_5) := (x_1x_2 + x_3x_4)(x_5 + c)$, and in Figure 4 a concrete example for computing this circuit.

- **Offline phase.** The trusted party $\mathcal{T}$ prepares the preprocessing data as follows.

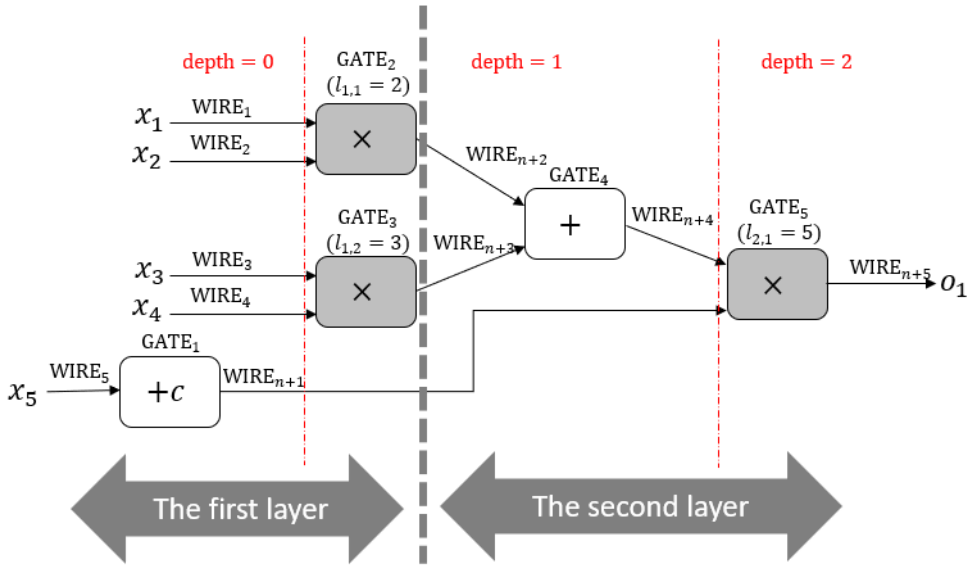
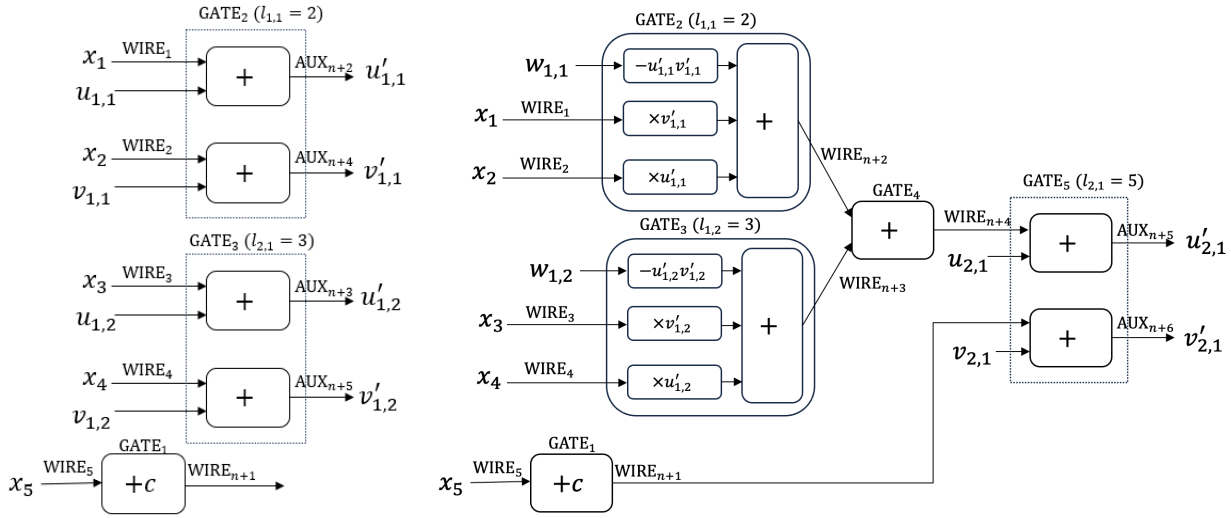


Figure 3: An arithmetic circuit computing $F(x_1, x_2, x_3, x_4, x_5) := (x_1x_2 + x_3x_4)(x_5 + c)$. Its multiplication depth is $D = 2$. The first layer contains $N_1 = 3$ gates, including $N_1^{\mathsf{mul}} = 2$ multiplication gates. The second layer contains two additional gates, including $N_2^{\mathsf{mul}} = 1$ multiplication gate, and hence the first two layers contain $N_2 = 5$ gates in total.

1. **Setup for the secret-sharing batches.** For every $\kappa \in [\Gamma]$, initialize the κ-th batch by
$$\{\mathsf{mvk}_{i_1,i_2}^{(\kappa)}\}_{i_1,i_2\in[n]} \leftarrow \Pi^{(\kappa)}.\mathsf{Setup}(1^\lambda).$$

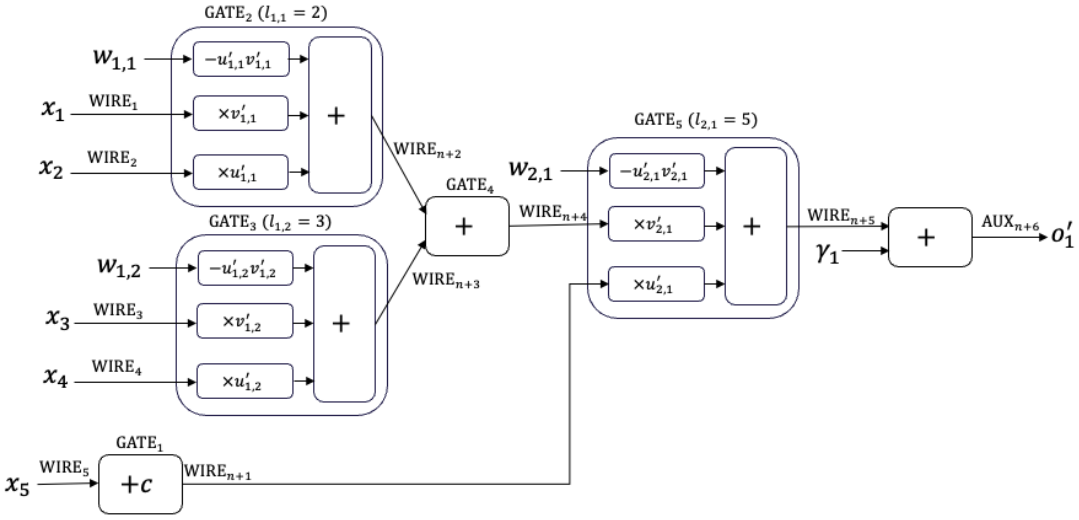
2. **Preparation for the inputs.** For every $i^* \in [n]$, sample a uniformly random input mask $\Delta_{i^*} \leftarrow \mathbb{Z}_q$ and privately send Δ_{i^*} to the input client $\mathcal{C}_{i^*}$. For every $\kappa \in [\Gamma]$, generate the i^*-th sharing in the κ-th batch by
$$(\{\mathsf{vk}_{i_1,i_2,i^*}^{(\kappa)}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}_{i,i^*}^{(\kappa)}\}_{i\in[n]}) \leftarrow \Pi^{(\kappa)}.\mathsf{Split}\big(i^*, \Delta_{i^*}, \{\mathsf{mvk}_{i_1,i_2}^{(\kappa)}\}_{i_1,i_2\in[n]}\big).$$

3. **Preparation for the gates.** For every $g \in [N]$, let $\iota := n + g$, and let ν be the unique layer containing GATE_g.
   - If GATE_g is linear, then for every $\kappa \in [\nu, \Gamma]$, generate
   $$(\{\mathsf{vk}_{i_1,i_2,\iota}^{(\kappa)}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}_{i,\iota}^{(\kappa)}\}_{i\in[n]}) \leftarrow \Pi^{(\kappa)}.\mathsf{Split}\big(\iota, 0, \{\mathsf{mvk}_{i_1,i_2}^{(\kappa)}\}_{i_1,i_2\in[n]}\big).$$
   - If GATE_g is a multiplication gate, let $\varsigma \in [N_\nu^{\mathsf{mul}}]$ be the unique integer such that $g = l_{\nu,\varsigma}$. Sample $u_{\nu,\varsigma}, v_{\nu,\varsigma} \leftarrow \mathbb{Z}_q$ and set $w_{\nu,\varsigma} := u_{\nu,\varsigma}v_{\nu,\varsigma}$. Define $\iota_u := n + g$ and $\iota_v := n + N_\nu + \varsigma$.
   Generate the ι_u-th and ι_v-th sharings in the ν-th batch by
   $$(\{\mathsf{vk}_{i_1,i_2,\iota_u}^{(\nu)}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}_{i,\iota_u}^{(\nu)}\}_{i\in[n]}) \leftarrow \Pi^{(\nu)}.\mathsf{Split}\big(\iota_u, u_{\nu,\varsigma}, \{\mathsf{mvk}_{i_1,i_2}^{(\nu)}\}_{i_1,i_2\in[n]}\big),$$
   and
   $$(\{\mathsf{vk}_{i_1,i_2,\iota_v}^{(\nu)}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}_{i,\iota_v}^{(\nu)}\}_{i\in[n]}) \leftarrow \Pi^{(\nu)}.\mathsf{Split}\big(\iota_v, v_{\nu,\varsigma}, \{\mathsf{mvk}_{i_1,i_2}^{(\nu)}\}_{i_1,i_2\in[n]}\big).$$



(a) The first subcircuit.

(b) The second subcircuit.



(c) The third subcircuit.

Figure 4: Decomposition of the circuit in Figure 3 into $\Gamma = D + 1 = 3$ subcircuits, each computing a linear function. For each circuit, we construct a corresponding secret-sharing batch with $M_1 = n + N_1 + N_1^{\mathsf{mul}} = n + 5$, $M_2 = n + N_2 + N_2^{\mathsf{mul}} = n + 6$, and $M_3 = n + N + n' = n + 6$ sharings, respectively.

For every $\kappa \in [\nu+1, \Gamma]$, generate a sharing of $w_{\nu,\varsigma}$ at coordinate $\iota_u = n+g$ by

$$(\{\mathsf{vk}^{(\kappa)}_{i_1,i_2,\iota_u}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}^{(\kappa)}_{i,\iota_u}\}_{i\in[n]}) \leftarrow \Pi^{(\kappa)}.\mathsf{Split}\big(\iota_u, w_{\nu,\varsigma}, \{\mathsf{mvk}^{(\kappa)}_{i_1,i_2}\}_{i_1,i_2\in[n]}\big).$$

4. **Preparation for the outputs.** For every $h \in [n']$, sample $\gamma_h \leftarrow \mathbb{Z}_q$ and let $\iota_h := n+N+h$. Generate the ι_h-th sharing in the final quantum batch by

$$(\{\mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota_h}\}_{i_1,i_2\in[n]}, \{\mathsf{sh}^{(\Gamma)}_{i,\iota_h}\}_{i\in[n]}) \leftarrow \Pi^{(\Gamma)}.\mathsf{Split}\big(\iota_h, \gamma_h, \{\mathsf{mvk}^{(\Gamma)}_{i_1,i_2}\}_{i_1,i_2\in[n]}\big).$$

   In addition, invoke $\langle\gamma_h\rangle \leftarrow \Pi_{\mathsf{ROpenSS}}.\mathsf{Split}(1^\lambda, \gamma_h)$.

5. **Packing.** For every $i_1 \in [n]$, let

$$\mathsf{pdata}_{i_1} := \left\{\mathsf{sh}^{(\kappa)}_{i_1,\iota_\kappa}\right\}_{\kappa\in[\Gamma],\,\iota_\kappa\in[M_\kappa]},$$

   and let

$$\mathsf{ver_key}_{i_1} := \left(\{\mathsf{mvk}^{(\kappa)}_{i_1,i_2}\}_{i_2\in[n],\,\kappa\in[\Gamma]}, \{\mathsf{vk}^{(\kappa)}_{i_1,i_2,\iota_\kappa}\}_{i_2\in[n],\,\kappa\in[\Gamma],\,\iota_\kappa\in[M_\kappa]}\right).$$

   Give $(\mathsf{pdata}_{i_1}, \mathsf{ver_key}_{i_1})$ to $\mathcal{P}_{i_1}$. The local shares generated by $\{\langle\gamma_h\rangle\}_{h\in[n']}$ are distributed among $\mathcal{P}_1, \ldots, \mathcal{P}_n$ according to Π_{ROpenSS}.

6. Initialize the adaptive deletion set $\mathcal{DS} \leftarrow \emptyset$.

7. Through out the protocol, we parse $\mathsf{sh}^{(\kappa)}_{i_1,\iota} = \left(|\psi^{(\kappa)}_{i_1,\iota}\rangle, \sigma^{(\kappa)}_\iota\right)$ for every $i_1 \in [n],\ \kappa \in [\Gamma],\ \iota \in [M_\kappa]$.[8]

▷ *For every $\kappa \in [D]$ and $\iota_\kappa \in [M_\kappa]$, define*

$$\mathsf{val}_{\kappa,\iota_\kappa} := \begin{cases} \Delta_{\iota_\kappa}, & \iota_\kappa \in [n], \\ 0, & \iota_\kappa = n+g,\ g \in [N_\kappa],\ \text{and } \mathsf{GATE}_g \text{ is linear}, \\ w_{\nu,\varsigma}, & \iota_\kappa = n+l_{\nu,\varsigma},\ \nu \in [\kappa-1],\ \varsigma \in [N^{\mathsf{mul}}_\nu], \\ u_{\kappa,\varsigma}, & \iota_\kappa = n+l_{\kappa,\varsigma},\ \varsigma \in [N^{\mathsf{mul}}_\kappa], \\ v_{\kappa,\varsigma}, & \iota_\kappa = n+N_\kappa+\varsigma,\ \varsigma \in [N^{\mathsf{mul}}_\kappa]. \end{cases}$$

For the final batch, define

$$\mathsf{val}_{\Gamma,\iota_\Gamma} := \begin{cases} \Delta_{\iota_\Gamma}, & \iota_\Gamma \in [n], \\ 0, & \iota_\Gamma = n+g,\ g \in [N],\ \text{and } \mathsf{GATE}_g \text{ is linear}, \\ w_{\nu,\varsigma}, & \iota_\Gamma = n+l_{\nu,\varsigma},\ \nu \in [D],\ \varsigma \in [N^{\mathsf{mul}}_\nu], \\ \gamma_h, & \iota_\Gamma = n+N+h,\ h \in [n']. \end{cases}$$

Thus, the offline phase distributes

$$\{[\![\mathsf{val}_{\kappa,\iota_\kappa}]\!]_{\kappa,\iota_\kappa}\}_{\kappa\in[\Gamma],\,\iota_\kappa\in[M_\kappa]} \quad \text{and} \quad \{\langle\gamma_h\rangle\}_{h\in[n']}$$

among $\mathcal{P}_1, \ldots, \mathcal{P}_n$.

[8] Every party wishing to recover the correct output, including a party whose quantum shares have been deleted, must keep the public classical masks updated according to the prescribed operations.

- **Input phase.** The input clients and computing parties proceed as follows:
  1. For every $i^* \in [n]$, client $\mathcal{C}_{i^*}$ broadcasts $\Delta'_{i^*} := x_{i^*} - \Delta_{i^*}$ and retires.
  2. For every $i_1 \in [n]$ and every $i^* \in [n]$, upon receiving Δ'_{i^*}, if $i_1 \notin \mathcal{DS}$, computing party $\mathcal{P}_{i_1}$ computes, for every $\kappa \in [\Gamma]$,
  $$\widetilde{\mathsf{sh}}^{(\kappa)}_{i_1,i^*} := \Pi^{(\kappa)}.\mathsf{AddSh}\big(\Delta'_{i^*}, i_1, i^*, \mathsf{sh}^{(\kappa)}_{i_1,i^*}\big),$$
  and replaces $\mathsf{sh}^{(\kappa)}_{i_1,i^*}$ by $\widetilde{\mathsf{sh}}^{(\kappa)}_{i_1,i^*}$.

▷ *After the input phase, for every $\kappa \in [\Gamma]$ and $i^* \in [n]$, the parties hold*
$$[\![x_{i^*}]\!]_{\kappa,i^*} \Leftarrow [\![\Delta_{i^*}]\!]_{\kappa,i^*} + \Delta'_{i^*}.$$

- **Evaluation phase.** The evaluation consists of Γ rounds $\mathsf{Round}_1, \ldots, \mathsf{Round}_\Gamma$. In $\mathsf{Round}_{\mathsf{cur}}$, where $\mathsf{cur} \in [\Gamma]$, the parties enumerate g from 1 to N_{cur} and evaluate GATE_g as follows.

  – **Scalar addition.** Suppose GATE_g is a c-sADD gate with input wire WIRE_{ι_1}, and let $\iota_0 := n + g$ be its output-wire label.
  For every $i_1 \in [n]$, party $\mathcal{P}_{i_1}$ proceeds as follows:
  1. For every $i_2 \in [n]$, compute
  $$(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}\big(1, i_1, i_2, \iota_1, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}\big).$$
  Replace the corresponding verification keys by the updated ones.
  2. If $i_1 \notin \mathcal{DS}$, compute
  $$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_1}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}\big(1, i_1, \iota_1, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_1}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_0}\big),$$
  and then compute
  $$\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0} := \Pi^{(\mathsf{cur})}.\mathsf{AddSh}\big(c, i_1, \iota_0, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}\big).$$
  Replace the corresponding shares by the updated ones.

  ▷ *The parties compute*
  $$[\![s + c]\!]_{\mathsf{cur},\iota_0} \Leftarrow [\![0]\!]_{\mathsf{cur},\iota_0} + [\![s]\!]_{\mathsf{cur},\iota_1} + c.$$

  – **Addition.** Suppose GATE_g is an ADD gate with input wires WIRE_{ι_1} and WIRE_{ι_2}, and let $\iota_0 := n + g$.
  For every $i_1 \in [n]$, party $\mathcal{P}_{i_1}$ proceeds as follows:
  1. For every $i_2 \in [n]$, compute
  $$(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \overline{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}\big(1, i_1, i_2, \iota_1, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}\big),$$
  and
  $$(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_2}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}\big(1, i_1, i_2, \iota_2, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_2}, \overline{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}\big).$$
  Replace the corresponding verification keys by the updated ones.

2. If $i_1 \notin \mathcal{DS}$, compute

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_1}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}\big(1, i_1, \iota_1, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_1}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_0}\big),$$

and then compute

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_2}, \widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}\big(1, i_1, \iota_2, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_2}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}\big).$$

Replace the corresponding shares by the updated ones.

▷ *The parties compute*

$$[\![s_1 + s_2]\!]_{\mathsf{cur},\iota_0} \Leftarrow [\![0]\!]_{\mathsf{cur},\iota_0} + [\![s_1]\!]_{\mathsf{cur},\iota_1} + [\![s_2]\!]_{\mathsf{cur},\iota_2}.$$

– **Scalar multiplication.** Suppose GATE_g is a c-sMUL gate with input wire WIRE_{ι_1}, and let $\iota_0 := n + g$.

For every $i_1 \in [n]$, party $\mathcal{P}_{i_1}$ proceeds as follows:

1. For every $i_2 \in [n]$, compute

$$\begin{aligned}&(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})\\ &:= \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}\big(c, i_1, i_2, \iota_1, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}\big).\end{aligned}$$

Replace the corresponding verification keys by the updated ones.

2. If $i_1 \notin \mathcal{DS}$, compute

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_1}, \widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}\big(c, i_1, \iota_1, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_1}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_0}\big).$$

Replace the corresponding shares by the updated ones.

▷ *The parties compute*

$$[\![cs]\!]_{\mathsf{cur},\iota_0} \Leftarrow [\![0]\!]_{\mathsf{cur},\iota_0} + c[\![s]\!]_{\mathsf{cur},\iota_1}.$$

– **Multiplication.** Suppose GATE_g is a multiplication gate.

- If $\mathsf{cur} \in [D]$ and $g = l_{\mathsf{cur},\varsigma}$ for some $\varsigma \in [N^{\mathsf{mul}}_{\mathsf{cur}}]$, let the input wires be WIRE_{ι_1} and WIRE_{ι_2}, and define $\iota_3 := n + g$ and $\iota_4 := n + N_{\mathsf{cur}} + \varsigma$.

  For every $i_1 \in [n]$, party $\mathcal{P}_{i_1}$ proceeds as follows:

  1. For every $b \in \{1, 2\}$ and $i_2 \in [n]$, compute

$$\begin{aligned}&(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_b}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_{b+2}})\\ &:= \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}\big(1, i_1, i_2, \iota_b, \iota_{b+2}, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_b}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_{b+2}}\big).\end{aligned}$$

  Replace the corresponding verification keys by the updated ones.

  2. If $i_1 \notin \mathcal{DS}$, for every $b \in \{1, 2\}$, compute

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_b}, \widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_{b+2}}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}\big(1, i_1, \iota_b, \iota_{b+2}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_b}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_{b+2}}\big).$$

  Replace the corresponding shares by the updated ones.

▷ *The parties compute*

$$[\![u'_{\mathsf{cur},\varsigma}]\!]_{\mathsf{cur},\iota_3} \Leftarrow [\![u_{\mathsf{cur},\varsigma}]\!]_{\mathsf{cur},\iota_3} + [\![s_1]\!]_{\mathsf{cur},\iota_1},$$

and

$$[\![v'_{\mathsf{cur},\varsigma}]\!]_{\mathsf{cur},\iota_4} \Leftarrow [\![v_{\mathsf{cur},\varsigma}]\!]_{\mathsf{cur},\iota_4} + [\![s_2]\!]_{\mathsf{cur},\iota_2}.$$

- Otherwise, there exist unique $\nu \in [\mathsf{cur}-1]$ and $\varsigma \in [N_\nu^{\mathsf{mul}}]$ such that $g = l_{\nu,\varsigma}$. Let $u'_{\nu,\varsigma}$ and $v'_{\nu,\varsigma}$ be the values reconstructed at the end of Round_ν. Let the input wires be WIRE_{ι_1} and WIRE_{ι_2}, and define $\iota_0 := n + g$.
  For every $i_1 \in [n]$, party $\mathcal{P}_{i_1}$ proceeds as follows:
  1. For every $i_2 \in [n]$, compute

$$\begin{aligned}&(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \overline{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})\\ &:= \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}\big(v'_{\nu,\varsigma}, i_1, i_2, \iota_1, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_1}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}\big),\end{aligned}$$

     and

$$\begin{aligned}&(\widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_2}, \widetilde{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0})\\ &:= \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddVk}\big(u'_{\nu,\varsigma}, i_1, i_2, \iota_2, \iota_0, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota_2}, \overline{\mathsf{vk}}^{(\mathsf{cur})}_{i_1,i_2,\iota_0}\big).\end{aligned}$$

     Replace the corresponding verification keys by the updated ones.
  2. If $i_1 \notin \mathcal{DS}$, compute

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_1}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}\big(v'_{\nu,\varsigma}, i_1, \iota_1, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_1}, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_0}\big),$$

     then

$$(\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_2}, \widehat{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}) := \Pi^{(\mathsf{cur})}.\mathsf{ScaleAddSh}\big(u'_{\nu,\varsigma}, i_1, \iota_2, \iota_0, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota_2}, \overline{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}\big),$$

     and finally

$$\widetilde{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0} := \Pi^{(\mathsf{cur})}.\mathsf{AddSh}\big(-u'_{\nu,\varsigma}v'_{\nu,\varsigma}, i_1, \iota_0, \widehat{\mathsf{sh}}^{(\mathsf{cur})}_{i_1,\iota_0}\big).$$

     Replace the corresponding shares by the updated ones.

▷ *The parties compute*

$$[\![s_1 s_2]\!]_{\mathsf{cur},\iota_0} \Leftarrow [\![w_{\nu,\varsigma}]\!]_{\mathsf{cur},\iota_0} + v'_{\nu,\varsigma}[\![s_1]\!]_{\mathsf{cur},\iota_1} + u'_{\nu,\varsigma}[\![s_2]\!]_{\mathsf{cur},\iota_2} - u'_{\nu,\varsigma}v'_{\nu,\varsigma}.$$

– **End-of-round reconstruction.** If $\mathsf{cur} \in [D]$, define

$$\mathsf{OPEN}_{\mathsf{cur}} := \Big\{n + l_{\mathsf{cur},\varsigma},\ n + N_{\mathsf{cur}} + \varsigma : \varsigma \in [N^{\mathsf{mul}}_{\mathsf{cur}}]\Big\},\ \mathsf{DEL}_{\mathsf{cur}} := [M_{\mathsf{cur}}] \setminus \mathsf{OPEN}_{\mathsf{cur}}. \quad (35)$$

For every $i_1 \in [n]$, party $\mathcal{P}_{i_1}$ proceeds as follows:

1. If $i_1 \notin \mathcal{DS}$, for every $\iota \in \mathsf{DEL}_{\mathsf{cur}}$, compute

$$\mathsf{cert}^{(\mathsf{cur})}_{i_1,\iota} \leftarrow \Pi^{(\mathsf{cur})}.\mathsf{Delete}\big(i_1, \iota, \mathsf{sh}^{(\mathsf{cur})}_{i_1,\iota}\big),$$

   and broadcast

$$\mathsf{rec_cert}^{(\mathsf{cur})}_{i_1} := \{\mathsf{cert}^{(\mathsf{cur})}_{i_1,\iota}\}_{\iota \in \mathsf{DEL}_{\mathsf{cur}}}.$$

2. After receiving $\mathsf{rec_cert}^{(\mathsf{cur})}_{i_2}$ from every $\mathcal{P}_{i_2}$ such that $i_2 \notin \mathcal{DS}$, verify

$$\Pi^{(\mathsf{cur})}.\mathsf{Verify}\big(i_1, i_2, \iota, \mathsf{mvk}^{(\mathsf{cur})}_{i_1,i_2}, \mathsf{vk}^{(\mathsf{cur})}_{i_1,i_2,\iota}, \mathsf{cert}^{(\mathsf{cur})}_{i_2,\iota}\big) = \top$$

for every $i_2 \notin \mathcal{DS}$ and $\iota \in \mathsf{DEL}_{\mathsf{cur}}$. Broadcast $\top$ if every verification accepts, and broadcast $\bot$ otherwise.

3. Abort immediately upon receiving $\bot$ from any party. Otherwise, after receiving $\top$ from every party, if $i_1 \notin \mathcal{DS}$, measure

$$\left\{\left|\boldsymbol{\psi}^{(\mathsf{cur})}_{i_1,n+l_{\mathsf{cur},\varsigma}}\right\rangle, \left|\boldsymbol{\psi}^{(\mathsf{cur})}_{i_1,n+N_{\mathsf{cur}}+\varsigma}\right\rangle\right\}_{\varsigma\in[N^{\mathsf{mul}}_{\mathsf{cur}}]}$$

in the computational basis, obtaining

$$\left\{\mathbf{z}^{(\mathsf{cur})}_{i_1,\varsigma,1}, \mathbf{z}^{(\mathsf{cur})}_{i_1,\varsigma,2}\right\}_{\varsigma\in[N^{\mathsf{mul}}_{\mathsf{cur}}]},$$

and broadcast the measurement results.

4. After receiving the corresponding measurement results from every undeleted party, apply polynomial interpolation with error correction (Lemma 8) to the reconstruction strings for $u'_{\mathsf{cur},\varsigma}$ and $v'_{\mathsf{cur},\varsigma}$ for every $\varsigma \in [N^{\mathsf{mul}}_{\mathsf{cur}}]$. Let $\widehat{f}^{(\mathsf{cur})}_{\varsigma,1}$ and $\widehat{f}^{(\mathsf{cur})}_{\varsigma,2}$ denote the degree-at-most-d polynomials returned by the two decoders. Abort if any decoder outputs $\bot$. Otherwise, for every $\varsigma \in [N^{\mathsf{mul}}_{\mathsf{cur}}]$, $i_2 \in [n] \setminus \mathcal{DS}$, and $b \in \{1,2\}$, check whether

$$\left|\left\{j \in [m] : \mathbf{z}^{(\mathsf{cur})}_{i_2,\varsigma,b}[j] \neq \widehat{f}^{(\mathsf{cur})}_{\varsigma,b}((i_2-1)m+j)\right\}\right| \leq n\ell.$$

Abort if any check fails. Otherwise, for every $\varsigma \in [N^{\mathsf{mul}}_{\mathsf{cur}}]$, reconstruct $u'_{\mathsf{cur},\varsigma}$ from $\widehat{f}^{(\mathsf{cur})}_{\varsigma,1}$ and $\sigma^{(\mathsf{cur})}_{n+l_{\mathsf{cur},\varsigma}}$, and reconstruct $v'_{\mathsf{cur},\varsigma}$ from $\widehat{f}^{(\mathsf{cur})}_{\varsigma,2}$ and $\sigma^{(\mathsf{cur})}_{n+N_{\mathsf{cur}}+\varsigma}$.

▷ *At the end of* $\mathsf{Round}_{\mathsf{cur}}$*, for every* $\varsigma \in [N^{\mathsf{mul}}_{\mathsf{cur}}]$*, the parties reconstruct*

$$u'_{\mathsf{cur},\varsigma} = u_{\mathsf{cur},\varsigma} + s_{\mathsf{cur},\varsigma,1}, \qquad v'_{\mathsf{cur},\varsigma} = v_{\mathsf{cur},\varsigma} + s_{\mathsf{cur},\varsigma,2},$$

while deleting all other sharings in the cur*-th batch.*

– **Output masking in the final round.** In Round_Γ, after all gates have been evaluated, let the output wires be $\mathsf{WIRE}_{\iota^{(o)}_1}, \ldots, \mathsf{WIRE}_{\iota^{(o)}_{n'}}$. For every $h \in [n']$, define $\iota_h := n+N+h$.

For every $i_1 \in [n]$, party $\mathcal{P}_{i_1}$ proceeds as follows:

1. For every $i_2 \in [n]$ and $h \in [n']$, compute

$$\begin{aligned}&(\widetilde{\mathsf{vk}}^{(\Gamma)}_{i_1,i_2,\iota^{(o)}_h}, \widetilde{\mathsf{vk}}^{(\Gamma)}_{i_1,i_2,\iota_h})\\ &:= \Pi^{(\Gamma)}.\mathsf{ScaleAddVk}\big(1, i_1, i_2, \iota^{(o)}_h, \iota_h, \mathsf{mvk}^{(\Gamma)}_{i_1,i_2}, \mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota^{(o)}_h}, \mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota_h}\big).\end{aligned}$$

Replace the corresponding verification keys by the updated ones.

2. If $i_1 \notin \mathcal{DS}$, for every $h \in [n']$, compute

$$(\widetilde{\mathsf{sh}}^{(\Gamma)}_{i_1,\iota^{(o)}_h}, \widetilde{\mathsf{sh}}^{(\Gamma)}_{i_1,\iota_h}) := \Pi^{(\Gamma)}.\mathsf{ScaleAddSh}\big(1, i_1, \iota^{(o)}_h, \iota_h, \mathsf{sh}^{(\Gamma)}_{i_1,\iota^{(o)}_h}, \mathsf{sh}^{(\Gamma)}_{i_1,\iota_h}\big).$$

Replace the corresponding shares by the updated ones.

▷ *If the h-th circuit output is o_h, then the parties compute*

$$[\![o'_h]\!]_{\Gamma,\iota_h} := [\![o_h + \gamma_h]\!]_{\Gamma,\iota_h} \Leftarrow [\![o_h]\!]_{\Gamma,\iota_h^{(o)}} + [\![\gamma_h]\!]_{\Gamma,\iota_h}.$$

- **Adaptive deletion for $\mathcal{P}_{i_1}$.** At an adaptive-query point before $\mathsf{Round}_{\mathsf{cur}}$ (we regard the initial adaptive-query point before the input phase as the adaptive-query point before Round_1), when the adversary instructs $\mathcal{P}_{i_1}$ to delete its remaining quantum state, the parties proceed as follows:

  1. For every $\kappa \in [\mathsf{cur}, \Gamma]$ and $\iota_\kappa \in [M_\kappa]$, party $\mathcal{P}_{i_1}$ computes

  $$\mathsf{cert}^{(\kappa)}_{i_1,\iota_\kappa} \leftarrow \Pi^{(\kappa)}.\mathsf{Delete}\big(i_1, \iota_\kappa, \mathsf{sh}^{(\kappa)}_{i_1,\iota_\kappa}\big),$$

  and broadcasts

  $$\mathsf{del_cert}_{i_1} := \left\{\mathsf{cert}^{(\kappa)}_{i_1,\iota_\kappa}\right\}_{\kappa\in[\mathsf{cur},\Gamma],\, \iota_\kappa\in[M_\kappa]}.$$

  2. Every $\mathcal{P}_{i_2}$ verifies each certificate using the corresponding verification key. It broadcasts $\top$ if all verifications accept, and broadcasts $\bot$ otherwise.
  3. Every party aborts upon receiving $\bot$ from any party. Otherwise, update $\mathcal{DS} \leftarrow \mathcal{DS} \cup \{i_1\}$.

▷ *The deletion phase erases all remaining quantum secret-sharing batches held by $\mathcal{P}_{i_1}$.*

- **Output phase.** For every $h \in [n']$, let $\iota_h := n + N + h$. For every $i_1 \in [n]$, party $\mathcal{P}_{i_1}$ proceeds as follows:

  1. If $i_1 \notin \mathcal{DS}$, for every $\iota \in [n+N]$, compute

  $$\mathsf{cert}^{(\Gamma)}_{i_1,\iota} \leftarrow \Pi^{(\Gamma)}.\mathsf{Delete}\big(i_1, \iota, \mathsf{sh}^{(\Gamma)}_{i_1,\iota}\big),$$

  and broadcast

  $$\mathsf{rec_cert}^{(\Gamma)}_{i_1} := \left\{\mathsf{cert}^{(\Gamma)}_{i_1,\iota}\right\}_{\iota\in[n+N]}.$$

  2. After receiving $\mathsf{rec_cert}^{(\Gamma)}_{i_2}$ from every $\mathcal{P}_{i_2}$ such that $i_2 \notin \mathcal{DS}$, verify

  $$\Pi^{(\Gamma)}.\mathsf{Verify}\big(i_1, i_2, \iota, \mathsf{mvk}^{(\Gamma)}_{i_1,i_2}, \mathsf{vk}^{(\Gamma)}_{i_1,i_2,\iota}, \mathsf{cert}^{(\Gamma)}_{i_2,\iota}\big) = \top$$

  for every $i_2 \notin \mathcal{DS}$ and $\iota \in [n+N]$. Broadcast $\top$ if every verification accepts, and broadcast $\bot$ otherwise.

  3. Abort immediately upon receiving $\bot$ from any party. Otherwise, after receiving $\top$ from every party, if $i_1 \notin \mathcal{DS}$, measure

  $$\left\{\left|\boldsymbol{\psi}^{(\Gamma)}_{i_1,\iota_h}\right\rangle\right\}_{h\in[n']}$$

  in the computational basis, obtaining $\{\mathbf{z}^{(\Gamma)}_{i_1,\iota_h}\}_{h\in[n']}$, and broadcast these measurement results.

4. After receiving the corresponding measurement results from every undeleted party, apply polynomial interpolation with error correction (Lemma 8) to

$$\left\{\mathbf{z}^{(\Gamma)}_{i_2,\iota_h}\right\}_{i_2\in[n]\setminus\mathcal{DS}}$$

for every $h \in [n']$. Let $\widehat{f}^{(\Gamma)}_{\iota_h}$ denote the degree-at-most-d polynomial returned by the decoder for the h-th reconstruction. Abort if any decoder outputs $\perp$. Otherwise, for every $h \in [n']$ and $i_2 \in [n] \setminus \mathcal{DS}$, check whether

$$\left|\left\{j \in [m] : \mathbf{z}^{(\Gamma)}_{i_2,\iota_h}[j] \neq \widehat{f}^{(\Gamma)}_{\iota_h}((i_2-1)m+j)\right\}\right| \leq n\ell.$$

Abort if any check fails. Otherwise, reconstruct o'_h from $\widehat{f}^{(\Gamma)}_{\iota_h}$ and $\sigma^{(\Gamma)}_{\iota_h}$ for every $h \in [n']$.

5. For every $h \in [n']$, invoke $\gamma_h \leftarrow \Pi_{\mathsf{ROpenSS}}.\mathsf{Open}(\langle\gamma_h\rangle)$. Abort if any invocation returns $\perp$; otherwise, compute $o_h := o'_h - \gamma_h$.
6. If no abort occurs, every party outputs $(o_1, \ldots, o_{n'})$.

▷ *In the output phase, the parties delete every sharing in the final batch except* $\{[\![o'_h]\!]_{\Gamma,n+N+h}\}_{h\in[n']}$*, reconstruct the masked outputs* $o'_h = o_h + \gamma_h$*, robustly open the classical sharings* $\{\langle\gamma_h\rangle\}_{h\in[n']}$ *using* Π_{ROpenSS}*, and recover* $o_h = o'_h - \gamma_h$ *for every* $h \in [n']$*.*

Compared with the protocol in Section 5.2, the above construction uses only $D+1$ quantum secret-sharing batches rather than $N^{\mathsf{mul}}+1$ batches. The reconstruction correctness follows from the same argument as Lemma 36, applied independently to every reconstructed sharing. The security proof follows the proof in Section 5.4: the only difference is that all masked values associated with multiplication gates at the same multiplication depth are reconstructed simultaneously from one batch. The independent output masks ensure that only uniformly masked output sharings are reconstructed from the final quantum batch before the classical masks are robustly opened.

Theorem 48. *Suppose that* Π_{ROpenSS} *is an adaptively programmable* n*-out-of-*n *classical secret-sharing protocol with robust opening. Then the multiplication-depth-based protocol* $\widetilde{\Pi}^{\mathsf{MPC}}_F$ *securely realizes* F *in the outsourced-MPC model of Section 3.2, with adaptive semi-honest corruption of at most* $n-1$ *clients,* (t,n)*-adaptively secure unanimously verifiable deletion, and post-execution full corruption of the computing parties.*

Proof. The proof follows that of Theorem 37. The only difference is that all multiplication gates of the same multiplication depth are evaluated using one secret-sharing batch. We define the following experiments exactly as in Section 5.4, replacing the original multiplication batches by the depth-based batches.

- $\mathsf{Hyb}_0(F, \mathbf{x})$ is the real-world execution of $\widetilde{\Pi}^{\mathsf{MPC}}_F$.
- $\mathsf{Hyb}_1(F, \mathbf{x})$ changes the sampling order of the input masks: the challenger first samples each public input message $\Delta'_i \leftarrow \mathbb{Z}_q$ and then sets $\Delta_i := x_i - \Delta'_i$. This does not change the joint distribution.

- $\mathsf{Hyb}_2(F, \mathbf{x})$ obtains the protocol output from the ideal functionality and uses the programmable-completion simulator of Π_{ROpenSS} to program the classical output-mask sharings, exactly as in Section 5.4.

- $\mathsf{Sim}_2(F, \mathbf{x})$ is the corresponding ideal-world experiment in which the simulator samples the clients' public input messages uniformly, learns x_i whenever $\mathcal{C}_i$ is adaptively corrupted and simulates the corresponding client view as in Section 5.4, and replaces every pre-evaluation secret value underlying the quantum secret-sharing batches by an independent uniform value.

To compare $\mathsf{Hyb}_2(F, \mathbf{x})$ and $\mathsf{Sim}_2(F, \mathbf{x})$, we introduce the same auxiliary experiments as in Section 5.4.

- $\mathsf{Hyb}_3(F, \mathbf{x})$ and $\mathsf{Sim}_3(F, \mathbf{x})$ generate each batch by first sampling its post-evaluation description and then applying the inverse evaluation transformation.

- $\mathsf{Hyb}_4(F, \mathbf{x})$ and $\mathsf{Sim}_4(F, \mathbf{x})$ replace honest deletion measurements by simulated deletion certificates.

- $\mathsf{Hyb}_5(F, \mathbf{x})$ and $\mathsf{Sim}_5(F, \mathbf{x})$ generate the polynomial evaluations and reconstruction messages lazily by interpolation.

- $\mathsf{Hyb}_6(F, \mathbf{x})$ and $\mathsf{Sim}_6(F, \mathbf{x})$ purify the quantum shares delivered to the adversary.

- $\{\mathsf{Hyb}_{7,\tau'}(F, \mathbf{x}), \mathsf{Sim}_{7,\tau'}(F, \mathbf{x})\}_{\tau' \in [n]}$ introduce the additional global-projector measurements at the first τ' corrupt-then-delete events, with the same conditions, abort rules, and updates as in Section 5.4.

- $\{\mathsf{Hyb}_{8,\tau'}(F, \mathbf{x}), \mathsf{Sim}_{8,\tau'}(F, \mathbf{x})\}_{\tau' \in [n]}$ replace polynomial interpolation of the R data registers in coordinates DEL_κ (defined as in Eqn. (35) for $\kappa \in [D]$, and with $\mathsf{DEL}_\Gamma := [n + N]$ for $\kappa = \Gamma$) by fresh EPR pairs, and omit the corresponding constant terms and private registers storing them. The remaining coordinates retain their polynomial generation. The inverse evaluation transformations and the local/global projection checks follow Section 5.4.

The proofs of all transitions are the same as in Section 5.4, applied to each depth-based batch. Simultaneously reconstructing all masked values at one multiplication depth causes no additional difficulty, since their fresh independent masks make the reconstructed vector uniform; the same round-by-round argument as in Lemma 47 then applies. Moreover, all sharings in the same batch use the same checking positions, and the generalized sampling argument in Remark 35 applies to each post-evaluation sharing coordinate exactly as in Lemma 46. Hence the same hidden-checking-position and high-rate quantum-extractor arguments apply. After these replacements, the multiplication-round transcripts, masked-output reconstruction, programmable completion, abort behavior, and residual states revealed after finalization have the same distributions as in the ideal-world experiment. Therefore, the real-world and ideal-world executions are statistically indistinguishable, which proves the theorem. □

Acknowledgments

We thank Xiaoyu Ji and Yifan Song for their helpful guidance on MPC. We also thank ChatGPT for helping us polish the language and fix bugs in earlier versions of our security proof. Finally, we thank the anonymous reviewers for their valuable comments and suggestions.

References

[AHH24] Prabhanjan Ananth, Zihan Hu, and Zikuan Huang. Quantum key-revocable dual-regev encryption, revisited. In *TCC (3)*, volume 15366 of *Lecture Notes in Computer Science*, pages 257–288. Springer, 2024.

[AK25] Kasra Abbaszadeh and Jonathan Katz. Non-interactive zero-knowledge arguments with certified deletion. In *ASIACRYPT (8)*, volume 16252 of *Lecture Notes in Computer Science*, pages 381–410. Springer, 2025.

[AKN+23] Shweta Agrawal, Fuyuki Kitagawa, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Public key encryption with secure key leasing. In *EUROCRYPT (1)*, volume 14004 of *Lecture Notes in Computer Science*, pages 581–610. Springer, 2023.

[APV23] Prabhanjan Ananth, Alexander Poremba, and Vinod Vaikuntanathan. Revocable cryptography from learning with errors. In *TCC (4)*, volume 14372 of *Lecture Notes in Computer Science*, pages 93–122. Springer, 2023.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology—EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188. Springer, 2011.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.

[BGK+24] James Bartusek, Vipul Goyal, Dakshita Khurana, Giulio Malavolta, Justin Raizes, and Bhaskar Roberts. Software with certified deletion. In *EUROCRYPT (4)*, volume 14654 of *Lecture Notes in Computer Science*, pages 85–111. Springer, 2024.

[BI20] Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In *TCC (3)*, volume 12552 of *Lecture Notes in Computer Science*, pages 92–122. Springer, 2020.

[BK23] James Bartusek and Dakshita Khurana. Cryptography with certified deletion. In *CRYPTO (5)*, volume 14085 of *Lecture Notes in Computer Science*, pages 192–223. Springer, 2023.

[BKM+23] James Bartusek, Dakshita Khurana, Giulio Malavolta, Alexander Poremba, and Michael Walter. Weakening assumptions for publicly-verifiable deletion. In *TCC (4)*, volume 14372 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2023.

[BKP23] James Bartusek, Dakshita Khurana, and Alexander Poremba. Publicly-verifiable deletion via target-collapsing functions. In *CRYPTO (5)*, volume 14085 of *Lecture Notes in Computer Science*, pages 99–128. Springer, 2023.

[BN24] Dominique Bazin and Ryo Nishimaki. A note on security definitions for secret sharing with certified deletion. *IACR Cryptol. ePrint Arch.*, 2024:1690, 2024.

[BR24] James Bartusek and Justin Raizes. Secret sharing with certified deletion. In *CRYPTO (7)*, volume 14926 of *Lecture Notes in Computer Science*, pages 184–214. Springer, 2024.

[CGJL25] Orestis Chardouvelis, Vipul Goyal, Aayush Jain, and Jiahui Liu. Quantum key leasing for PKE and FHE with a classical lessor. In *EUROCRYPT (3)*, volume 15603 of *Lecture Notes in Computer Science*, pages 248–277. Springer, 2025.

[HKM+24] Taiga Hiroka, Fuyuki Kitagawa, Tomoyuki Morimae, Ryo Nishimaki, Tapas Pal, and Takashi Yamakawa. Certified everlasting secure collusion-resistant functional encryption, and more. In *EUROCRYPT (3)*, volume 14653 of *Lecture Notes in Computer Science*, pages 434–456. Springer, 2024.

[HMNY21] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. In *ASIACRYPT (1)*, volume 13090 of *Lecture Notes in Computer Science*, pages 606–636. Springer, 2021.

[HMNY22] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting zero-knowledge proof for QMA. In *CRYPTO (1)*, volume 13507 of *Lecture Notes in Computer Science*, pages 239–268. Springer, 2022.

[KMY25] Fuyuki Kitagawa, Tomoyuki Morimae, and Takashi Yamakawa. A simple framework for secure key leasing. In *EUROCRYPT (3)*, volume 15603 of *Lecture Notes in Computer Science*, pages 217–247. Springer, 2025.

[KN22] Fuyuki Kitagawa and Ryo Nishimaki. Functional encryption with secure key leasing. In *ASIACRYPT (4)*, volume 13794 of *Lecture Notes in Computer Science*, pages 569–598. Springer, 2022.

[KNP25] Fuyuki Kitagawa, Ryo Nishimaki, and Nikhil Pappu. PKE and ABE with collusion-resistant secure key leasing. In *CRYPTO (3)*, volume 16002 of *Lecture Notes in Computer Science*, pages 35–68. Springer, 2025.

[KNP26] Fuyuki Kitagawa, Ryo Nishimaki, and Nikhil Pappu. Collusion-resistant quantum secure key leasing beyond decryption. In *EUROCRYPT (1)*, volume 16541 of *Lecture Notes in Computer Science*, pages 571–602. Springer, 2026.

[KNY21] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. In *TCC (1)*, volume 13042 of *Lecture Notes in Computer Science*, pages 31–61. Springer, 2021.

[KNY23] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Publicly verifiable deletion from minimal assumptions. In *TCC (4)*, volume 14372 of *Lecture Notes in Computer Science*, pages 228–245. Springer, 2023.

[KS25] Jonathan Katz and Benjamin Sela. Secret sharing with publicly verifiable deletion. In *EUROCRYPT (3)*, volume 15603 of *Lecture Notes in Computer Science*, pages 278–304. Springer, 2025.

[KT23] Srijita Kundu and Ernest Y.-Z. Tan. Composably secure device-independent encryption with certified deletion. *Quantum*, 7:1047, 2023.

[MPY24] Tomoyuki Morimae, Alexander Poremba, and Takashi Yamakawa. Revocable quantum digital signatures. In *TQC*, volume 310 of *LIPIcs*, pages 5:1–5:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[Por23] Alexander Poremba. Quantum proofs of deletion for learning with errors. In *ITCS*, volume 251 of *LIPIcs*, pages 90:1–90:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[PWYZ24] Duong Hieu Phan, Weiqiang Wen, Xingyu Yan, and Jinwei Zheng. Adaptive hardcore bit and quantum key leasing over classical channel from LWE with polynomial modulus. In *ASIACRYPT (9)*, volume 15492 of *Lecture Notes in Computer Science*, pages 185–214. Springer, 2024.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Unr15] Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6):49:1–49:76, 2015.

[Win99] Andreas J. Winter. Coding theorem and strong converse for quantum channels. *IEEE Trans. Inf. Theory*, 45(7):2481–2485, 1999.

A Applications of the Linearly Homomorphic Structure of BB84-Type States

This appendix illustrates applications of the linear-homomorphic structure of BB84-type states developed in Section 4.1 beyond secret sharing. Section A.1 recalls the definition and security of classical linearly homomorphic secret-key encryption. Section A.2 defines linearly homomorphic secret-key encryption with certified deletion and its security requirements. Section A.3 then applies our BB84-based linear-homomorphic structure to construct such a scheme, denoted by $\Pi_{\mathsf{LHSKE\text{-}CD}}$. Finally, Section A.4 discusses how the same technique may be extended to the BB84-based certified-deletion compilers of [BK23].

A.1 Definition of Classical Linearly Homomorphic Secret-Key Encryption

Definition 49 (Linearly Homomorphic Secret-Key Encryption)**.** *A linearly homomorphic secret-key encryption consists of the following phases.*

- *Let $\mathbb{F}$ be a field denoting the message space. Let $\mathcal{SK}$ be the secret key space and $\mathcal{CT}$ be the ciphertext space.*

- $\mathsf{Gen}(1^\lambda)$: *This function takes in the security parameter* λ*, and outputs a secret encryption/decryption key* $\mathsf{sk} \in \mathcal{SK}$.
- $\mathsf{Enc}(\mathsf{sk} \in \mathcal{SK}, \mathsf{msg} \in \mathbb{F})$: *This function takes in the secret encryption/decryption key* sk *and a message* $\mathsf{msg} \in \mathbb{F}$*, and outputs a ciphertext* $\mathsf{ct} \in \mathcal{CT}$.
- $\mathsf{EvalCt}(c \in \mathbb{F}, \mathsf{ct}_1 \in \mathcal{CT}, \mathsf{ct}_2 \in \mathcal{CT})$: *This function takes in a scalar* $c \in \mathbb{F}$*, and two ciphertexts* $\mathsf{ct}_1, \mathsf{ct}_2$*, and outputs a new ciphertext* $\widetilde{\mathsf{ct}} \in \mathcal{CT}$.
- $\mathsf{Dec}(\mathsf{sk} \in \mathcal{SK}, \mathsf{ct} \in \mathcal{CT})$: *This function takes in the secret encryption/decryption key* sk *and a ciphertext* ct*, and outputs a message* msg.

Definition 50 (Correctness)**.** *For any* $\lambda \in \mathbb{N}$*, a linearly homomorphic secret-key encryption* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{EvalCt}, \mathsf{Dec})$ *must satisfy the following.*

- ***(Decryption Correctness.)*** *Let* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$*. For any* $\mathsf{msg} \in \mathbb{F}$*, if* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{msg})$*, then* $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{msg}$.
- ***(Linear Homomorphism.)*** *For any* $\mathsf{sk} \in \mathcal{SK}$ *and* $\mathsf{msg} \in \mathbb{F}$*, let* $\mathcal{E}_{\mathsf{sk}}(\mathsf{msg}) \subseteq \mathcal{CT}$ *denote the space of all possible outputs of* $\mathsf{Enc}(\mathsf{sk}, \mathsf{msg})$*. For any* $c \in \mathbb{F}$*,* $\mathsf{msg}_1, \mathsf{msg}_2 \in \mathbb{F}$*, and* $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{CT}$ *such that*
$$\forall i \in \{1, 2\}, \mathsf{ct}_i \in \mathcal{E}_{\mathsf{sk}}(\mathsf{msg}_i),$$
it holds that
$$\mathsf{EvalCt}(c, \mathsf{ct}_1, \mathsf{ct}_2) \in \mathcal{E}_{\mathsf{sk}}(c \cdot \mathsf{msg}_1 + \mathsf{msg}_2).$$

Definition 51 (Semantic Security)**.** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{EvalCt}, \mathsf{Dec})$ *be a linearly homomorphic secret-key encryption scheme. For any* QPT *adversary* $\mathcal{A}$*, any* $M = \mathrm{poly}(\lambda)$*, we define the experiment* $\mathsf{Exp}^{\mathsf{sem}}_{\Pi,\mathcal{A},M,\lambda}(b)$ *as follows:*

1. *The adversary* $\mathcal{A}$ *submits two sets of challenge messages* $\{\mathsf{msg}_i^{(0)}\}_{i\in[M]}, \{\mathsf{msg}_i^{(1)}\}_{i\in[M]}$ *to the challenger.*
2. *The challenger computes* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$*. And for every* $i \in [M]$*, the challenger computes* $(\mathsf{ct}_i) \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{msg}_i^{(b)})$*. Finally, it sends* $\mathsf{ct}_1, \ldots, \mathsf{ct}_M$ *to* $\mathcal{A}$.
3. $\mathcal{A}$ *outputs a bit* $b' \in \{0, 1\}$.

We say Π *satisfies* M-semantic security *if for any* QPT *adversary* $\mathcal{A}$*, we have*
$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{sem}}_{\Pi,\mathcal{A},M,\lambda}(0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{sem}}_{\Pi,\mathcal{A},M,\lambda}(1) = 1\right]\right| \leq \mathrm{negl}(\lambda)$$

A.2 Definition of Linearly Homomorphic Symmetric-Key Encryption with Certified Deletion

Definition 52 (Linearly Homomorphic Secret-Key Encryption with Certified Deletion)**.** *A linearly homomorphic secret-key encryption with certified deletion consists of the following phases.*

- *Let* $\mathbb{F}$ *be a field denoting the message space. Let* $\mathcal{SK}$ *be the secret key space,* $\mathcal{CT}$ *be the ciphertext space, and* $\mathcal{VK}$ *be the verification key space.*

- $\mathsf{Gen}(1^\lambda)$: *This function takes in the security parameter* λ*, and outputs a secret encryption/decryption key* $\mathsf{sk} \in \mathcal{SK}$.
- $\mathsf{Enc}(\mathsf{sk} \in \mathcal{SK}, \mathsf{msg} \in \mathbb{F})$: *This function takes in the secret encryption/decryption key* sk *and a message* $\mathsf{msg} \in \mathbb{F}$*, and outputs a ciphertext* $\mathsf{ct} \in \mathcal{CT}$ *and a secret verification key* $\mathsf{vk} \in \mathcal{VK}$.
- $\mathsf{EvalCt}(c \in \mathbb{F}, \mathsf{ct}_1 \in \mathcal{CT}, \mathsf{ct}_2 \in \mathcal{CT})$: *This function takes in a scalar* $c \in \mathbb{F}$*, and two ciphertexts* $\mathsf{ct}_1, \mathsf{ct}_2$*, and outputs two new ciphertexts* $\widetilde{\mathsf{ct}}_1, \widetilde{\mathsf{ct}}_2 \in \mathcal{CT}$.
- $\mathsf{EvalVk}(c \in \mathbb{F}, \mathsf{vk}_1 \in \mathcal{VK}, \mathsf{vk}_2 \in \mathcal{VK})$: *This function takes in a scalar* $c \in \mathbb{F}$*, and two verification keys* $\mathsf{vk}_1, \mathsf{vk}_2$*, and outputs two new verification keys* $\widetilde{\mathsf{vk}}_1, \widetilde{\mathsf{vk}}_2 \in \mathcal{VK}$.
- $\mathsf{Dec}(\mathsf{sk} \in \mathcal{SK}, \mathsf{ct} \in \mathcal{CT})$: *This function takes in the secret encryption/decryption key* sk *and a ciphertext* ct*, and outputs a message* msg.
- $\mathsf{Del}(\mathsf{ct} \in \mathcal{CT})$: *This function takes in a ciphertext* ct*, and outputs a deletion certificate* cert.
- $\mathsf{Vrfy}(\mathsf{vk} \in \mathcal{VK}, \mathsf{cert})$: *This function takes in a verification key* vk*, and a deletion certificate* cert*, and outputs either* $\top$ *(indicating accept) or* $\bot$ *(indicating reject).*

Definition 53 (Correctness)**.** *For any* $\lambda \in \mathbb{N}$*, a linearly homomorphic secret-key encryption* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{EvalCt}, \mathsf{EvalVk}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ *with certified deletion must satisfy the following.*

- ***(Decryption Correctness.)*** *Let* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$*. For any* $\mathsf{msg} \in \mathbb{F}$*, if* $(\mathsf{ct}, \mathsf{vk}) \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{msg})$*, then* $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{msg}$.
- ***(Deletion Correctness.)*** *Let* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$*. For any* $\mathsf{msg} \in \mathbb{F}$*, if* $(\mathsf{ct}, \mathsf{vk}) \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{msg})$*, then* $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{Del}(\mathsf{ct})) = \top$.
- ***(Linear Homomorphism.)*** *For any* $\mathsf{sk} \in \mathcal{SK}$*,* $\mathsf{msg} \in \mathbb{F}$*, let* $\mathcal{E}_{\mathsf{sk}}(\mathsf{msg}) \subseteq \mathcal{CT} \times \mathcal{VK}$ *denote the space of all possible outputs of* $\mathsf{Enc}(\mathsf{sk}, \mathsf{msg})$*. For any* $c \in \mathbb{F}$*,* $\mathsf{msg}_1, \mathsf{msg}_2 \in \mathbb{F}$*,* $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{CT}$*, and* $\mathsf{vk}_1, \mathsf{vk}_2 \in \mathcal{VK}$ *such that*

$$\forall i \in \{1, 2\}, (\mathsf{ct}_i, \mathsf{vk}_i) \in \mathcal{E}_{\mathsf{sk}}(\mathsf{msg}_i),$$

let $(\widetilde{\mathsf{ct}}_1, \widetilde{\mathsf{ct}}_2) \leftarrow \mathsf{EvalCt}(c, \mathsf{ct}_1, \mathsf{ct}_2)$*, and* $(\widetilde{\mathsf{vk}}_1, \widetilde{\mathsf{vk}}_2) \leftarrow \mathsf{EvalVk}(c, \mathsf{vk}_1, \mathsf{vk}_2)$*. Then it holds that*

$$(\widetilde{\mathsf{ct}}_1, \widetilde{\mathsf{vk}}_1) \in \mathcal{E}_{\mathsf{sk}}(\mathsf{msg}_1), (\widetilde{\mathsf{ct}}_2, \widetilde{\mathsf{vk}}_2) \in \mathcal{E}_{\mathsf{sk}}(c \cdot \mathsf{msg}_1 + \mathsf{msg}_2).$$

Definition 54 (Semantic Security)**.** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{EvalCt}, \mathsf{EvalVk}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ *be a linearly homomorphic secret-key encryption with certified deletion. For any* QPT *adversary* $\mathcal{A}$*, any* $M = \mathrm{poly}(\lambda)$*, we define the experiment* $\mathsf{Exp}^{\mathsf{sem}}_{\Pi,\mathcal{A},M,\lambda}(b)$ *as follows:*

1. *The adversary* $\mathcal{A}$ *submits two sets of challenge messages* $\{\mathsf{msg}_i^{(0)}\}_{i\in[M]}, \{\mathsf{msg}_i^{(1)}\}_{i\in[M]}$ *to the challenger.*
2. *The challenger computes* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$*. And for every* $i \in [M]$*, the challenger computes* $(\mathsf{ct}_i, \mathsf{vk}_i) \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{msg}_i^{(b)})$*. Finally, it sends* $\mathsf{ct}_1, \ldots, \mathsf{ct}_M$ *to* $\mathcal{A}$.
3. $\mathcal{A}$ *outputs a bit* $b' \in \{0, 1\}$.

We say Π *satisfies* M-semantic security *if for any* QPT *adversary* $\mathcal{A}$*, we have*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{sem}}_{\Pi,\mathcal{A},M,\lambda}(0)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{sem}}_{\Pi,\mathcal{A},M,\lambda}(1)=1\right]\right|\leq \mathrm{negl}(\lambda)$$

Definition 55 (Certified Deletion Security)**.** *Let* $\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{EvalCt},\mathsf{EvalVk},\mathsf{Dec},\mathsf{Del},\mathsf{Vrfy})$ *be a linearly homomorphic secret-key encryption with certified deletion. For any* QPT *adversary* $\mathcal{A}$*, any* $M=\mathrm{poly}(\lambda)$*, and* $b\in\{0,1\}$*, we define the experiment* $\mathsf{Exp}^{\mathsf{cert\text{-}del}}_{\Pi,\mathcal{A},M,\lambda}(b)$*, as follows:*

1. *The adversary* $\mathcal{A}$ *submits to the challenger two sets of challenge messages* $\{\mathsf{msg}_i^{(0)}\}_{i\in[M]},\{\mathsf{msg}_i^{(1)}\}_{i\in[M]}$*, and an index set* $\mathcal{S}\subseteq[M]$ *such that* $\mathsf{msg}_i^{(0)}=\mathsf{msg}_i^{(1)}$ *for every* $i\in[M]\backslash\mathcal{S}$*.*
2. *The challenger computes* $\mathsf{sk}\leftarrow\mathsf{Gen}(1^\lambda)$*. And for every* $i\in[M]$*, the challenger computes* $(\mathsf{ct}_i,\mathsf{vk}_i)\leftarrow\mathsf{Enc}(\mathsf{sk},\mathsf{msg}_i^{(b)})$*. Finally it sends* $\mathsf{ct}_1,\ldots,\mathsf{ct}_M$ *to* $\mathcal{A}$*.*
3. *At some point,* $\mathcal{A}$ *sends deletion certificates* $\{\mathsf{cert}_i\}_{i\in\mathcal{S}}$ *and its residual state* ρ *to the challenger.*
4. *If* $\mathsf{Vrfy}(\mathsf{vk}_i,\mathsf{cert}_i)=\top$ *for every* $i\in S$*, then the challenger outputs* (sk,ρ)*; otherwise it outputs* $\bot$*.*

We say that Π *satisfies* M-certified deletion security *if for any* QPT *adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{TD}\left(\mathsf{Exp}^{\mathsf{cert\text{-}del}}_{\Pi,\mathcal{A},M,\lambda}(0),\mathsf{Exp}^{\mathsf{cert\text{-}del}}_{\Pi,\mathcal{A},M,\lambda}(1)\right)\leq\mathrm{negl}(\lambda).$$

A.3 Construction of Linearly Homomorphic Symmetric-Key Encryption with Certified Deletion

Assume the existence of a (quantum) semantically secure linearly homomorphic secret-key encryption scheme $\widehat{\Pi}$. We construct a linearly homomorphic secret-key encryption scheme $\Pi^{\mathsf{LHSKE-CD}}=(\mathsf{Gen},\mathsf{Enc},\mathsf{EvalCt},\mathsf{EvalVk},\mathsf{Dec},\mathsf{Del},\mathsf{Vrfy})$ with certified deletion as follows. The building blocks are as follows:

- Parameters: Let message space be $\mathbb{Z}_q$, where $q=2^{\Theta(\lambda)}$ is a prime. Let $m,M\in\mathrm{poly}(\lambda)$.
- $\mathsf{Gen}(1^\lambda)$: Sample $\widehat{\mathsf{sk}}\leftarrow\widehat{\Pi}.\mathsf{Gen}(1^\lambda)$ and $\boldsymbol{\theta}\leftarrow\{0,1\}^m$. Output $\mathsf{sk}:=(\widehat{\mathsf{sk}},\boldsymbol{\theta})$.
- $\mathsf{Enc}(\mathsf{sk},\mathsf{msg})$: Parse $\mathsf{sk}=(\widehat{\mathsf{sk}},\boldsymbol{\theta})$, sample $\mathbf{x}\leftarrow\mathbb{Z}_q^m$, and output $\mathsf{ct}:=(|\mathbf{x}^{\boldsymbol{\theta}}\rangle,\widehat{\Pi}.\mathsf{Enc}(\widehat{\mathsf{sk}},\mathsf{msg}+\sum_{i:\theta_i=0}x[i]))$ and $\mathsf{vk}:=(\mathbf{x},\boldsymbol{\theta})$.
- $\mathsf{Dec}(\mathsf{sk},\mathsf{ct})$: Parse $\mathsf{sk}=(\widehat{\mathsf{sk}},\boldsymbol{\theta})$ and $\mathsf{ct}=(|\psi\rangle,\widehat{\mathsf{ct}})$. Let $p:=\widehat{\Pi}.\mathsf{Dec}(\widehat{\mathsf{sk}},\widehat{\mathsf{ct}})$. Measure $|\psi\rangle$ on the computational basis[9], obtaining $z=(z_1,\ldots,z_m)\in\mathbb{Z}_q^m$, and output $\mathsf{msg}:=p-\sum_{i:\theta_i=0}z_i$.
- $\mathsf{Del}(\mathsf{ct})$: Parse $\mathsf{ct}=(|\psi\rangle,\widehat{\mathsf{ct}})$, measure the BB84 part $|\psi\rangle$ on the QFT basis, and output the resulting vector $\mathsf{cert}\in\mathbb{Z}_q^m$.
- $\mathsf{Vrfy}(\mathsf{vk},\mathsf{cert})$: Parse $\mathsf{vk}=(\mathbf{x},\boldsymbol{\theta})$, and output $\top$ iff $\mathsf{cert}[i]=x[i]$ for all i such that $\theta[i]=1$.

[9]If one does not want to destroy this state, he can measure $|\psi\rangle$ on the $\boldsymbol{\theta}$ basis, i.e., if $\theta[i]=0$, then measure the i-th register on the computational basis; otherwise, measure it on the QFT basis.

- $\mathsf{EvalCt}(c, \mathsf{ct}_1, \mathsf{ct}_2)$: parse $\mathsf{ct}_1 = (|\mathbf{x}_1^{\boldsymbol{\theta}}\rangle, \widehat{\mathsf{ct}}_1)$ and $\mathsf{ct}_2 = (|\mathbf{x}_2^{\boldsymbol{\theta}}\rangle, \widehat{\mathsf{ct}}_2)$, where both ciphertexts are generated under the same secret key and hence the same fixed basis string $\boldsymbol{\theta}$. Let $|\widetilde{\mathbf{x}}_1^{\boldsymbol{\theta}}\rangle|\widetilde{\mathbf{x}}_2^{\boldsymbol{\theta}}\rangle := \mathbf{U}_c^{\otimes m}|\mathbf{x}_1^{\boldsymbol{\theta}}\rangle|\mathbf{x}_2^{\boldsymbol{\theta}}\rangle$. Output $\widetilde{\mathsf{ct}}_1 := (|\widetilde{\mathbf{x}}_1^{\boldsymbol{\theta}}\rangle, \widehat{\mathsf{ct}}_1)$ and $\widetilde{\mathsf{ct}}_2 := (|\widetilde{\mathbf{x}}_2^{\boldsymbol{\theta}}\rangle, \widehat{\Pi}.\mathsf{EvalCt}(c, \widehat{\mathsf{ct}}_1, \widehat{\mathsf{ct}}_2))$.
- $\mathsf{EvalVk}(c, \mathsf{vk}_1, \mathsf{vk}_2)$: Parse $\mathsf{vk}_1 = (\mathbf{x}_1, \boldsymbol{\theta})$ and $\mathsf{vk}_2 = (\mathbf{x}_2, \boldsymbol{\theta})$. Let $\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2 \in \mathbb{Z}_q^m$ be two vectors defined as follows:

$$\forall i \in [m],\ \widetilde{x}_1[i] = \begin{cases} x_1[i], & \theta[i] = 0, \\ x_1[i] - cx_2[i], & \theta[i] = 1, \end{cases} \qquad \widetilde{x}_2[i] = \begin{cases} cx_1[i] + x_2[i], & \theta[i] = 0, \\ x_2[i], & \theta[i] = 1. \end{cases}$$

  Output $\widetilde{\mathsf{vk}}_1 := (\widetilde{\mathbf{x}}_1, \boldsymbol{\theta})$ and $\widetilde{\mathsf{vk}}_2 := (\widetilde{\mathbf{x}}_2, \boldsymbol{\theta})$.

The correctness (including linear homomorphism) of the scheme follows directly from construction and Theorem 23. The semantic security follows from the semantic security of the underlying SKE scheme $\widehat{\Pi}$. We only prove certified deletion security as follows.

Theorem 56. *When $m = \omega(M \log q + \lambda)$, $\Pi^{\mathsf{LHSKE-CD}}$ satisfies certified deletion security.*

Proof. We define a series of hybrids:

- $\mathsf{Hyb}_0(b)$: the certified deletion security experiment $\mathsf{Exp}^{\mathsf{cert\text{-}del}}_{\Pi,\mathcal{A},M,\lambda}(b)$:
- $\mathsf{Hyb}_1(b)$: this experiment is the same as $\mathsf{Hyb}_0(b)$ except that the challenger purify the BB84 state using EPR pairs. Concretely, for every $i \in [M]$, instead of giving $(|\mathbf{x}_i^{\boldsymbol{\theta}}\rangle, \widehat{\Pi}.\mathsf{Enc}(\widehat{\mathsf{sk}}, \mathsf{msg}_i^{(b)} + \sum_{j:\theta_j=0} x_i[j]))$ to $\mathcal{A}$, the challenger does the following:
  - For every $i \in [M]$, every $j \in [m]$, the challenger prepares an EPR pair on two registers $\mathsf{C}_{i,j}, \mathsf{A}_{i,j}$:
$$\sum_{x\in\mathbb{Z}_q} |x\rangle_{\mathsf{C}_{i,j}} \otimes |x\rangle_{\mathsf{A}_{i,j}} = \sum_{x\in\mathbb{Z}_q} \mathsf{QFT}^{-1}|x\rangle_{\mathsf{C}_{i,j}} \otimes \mathsf{QFT}|x\rangle_{\mathsf{A}_{i,j}}$$
    Let $\mathsf{C}_i := (\mathsf{C}_{i,1}, \ldots, \mathsf{C}_{i,m}), \mathsf{A}_i := (\mathsf{A}_{i,1}, \ldots, \mathsf{A}_{i,m})$.
  - Sample $\boldsymbol{\theta} \leftarrow \{0,1\}^m$. For every $i \in [M]$, measure C_i as follows to get $\mathbf{x}_i$:
    - if $\theta[j] = 0$, measure $\mathsf{C}_{i,j}$ on the computational basis to get $x_i[j]$.
    - if $\theta[j] = 1$, measure $\mathsf{C}_{i,j}$ on the inverse QFT basis to get $x_i[j]$.
  - For every $i \in [M]$, sample $r_i \leftarrow \mathbb{Z}_q$ and compute $\widehat{\mathsf{ct}}_i \leftarrow \widehat{\Pi}.\mathsf{Enc}(\widehat{\mathsf{sk}}, r_i)$.
  - Send $\{(\mathsf{A}_i, \widehat{\mathsf{ct}}_i)\}_{i\in[M]}$ to $\mathcal{A}$.
  - If $r_i = \mathsf{msg}_i^{(b)} + \sum_{j:\theta[j]=0} x_i[j]$ for every $i \in [M]$, proceed as in $\mathsf{Hyb}_0(b)$; otherwise output $\bot$ and abort.
- $\mathsf{Hyb}_2(b)$: This game is the same as $\mathsf{Hyb}_1(b)$ except that the challenger postpones the generation of $\boldsymbol{\theta}$ and the measurement of C_i to the time when it receives $\{\mathsf{cert}_i\}_{i\in\mathcal{S}}$ from $\mathcal{A}$.
- $\mathsf{Hyb}_3(b)$: this experiment is the same as $\mathsf{Hyb}_2(b)$ except that, after receiving $\{\mathsf{cert}_i\}_{i\in\mathcal{S}}$ from $\mathcal{A}$ but before measuring C_i according to $\boldsymbol{\theta}$, the challenger performs the following additional

measurement for every $i \in \mathcal{S}$. Define $\boldsymbol{\theta}_0 := \{j \in [m] : \theta[j] = 0\}$ and $\boldsymbol{\theta}_1 := \{j \in [m] : \theta[j] = 1\}$, and let $m' := |\boldsymbol{\theta}_0|$. For any $\mathbf{y} \in \mathbb{Z}_q^m$, define

$$\boldsymbol{\Pi}_{\mathbf{y}} := \Bigg(\sum_{\substack{\mathbf{u} \in \mathbb{Z}_q^{m'}: \\ \mathsf{HW}(\mathbf{u}) < (m'-1)/2}} (\mathsf{QFT}^{-1})^{\otimes m'} |\mathbf{u} + \mathbf{y}[\boldsymbol{\theta}_0]\rangle\langle \mathbf{u} + \mathbf{y}[\boldsymbol{\theta}_0]| \mathsf{QFT}^{\otimes m'} \Bigg) \otimes \Big((\mathsf{QFT}^{-1})^{\otimes |\boldsymbol{\theta}_1|} |\mathbf{y}[\boldsymbol{\theta}_1]\rangle\langle \mathbf{y}[\boldsymbol{\theta}_1]| \mathsf{QFT}^{\otimes |\boldsymbol{\theta}_1|} \Big).$$

The challenger performs the two-outcome measurement $\{\boldsymbol{\Pi}_{\mathsf{cert}_i}, \mathbf{I} - \boldsymbol{\Pi}_{\mathsf{cert}_i}\}$ on C_i. If the result is reject, the challenger outputs $\perp$ and aborts.

To prove the security of our scheme, we need to establish the indistinguishability of $\mathsf{Hyb}_0(0)$ and $\mathsf{Hyb}_0(1)$. We prove this by gathering up the lemmas about indistinguishablity between adjacent hybrids as follows.

$$\begin{aligned}
&\mathsf{TD}(\mathsf{Hyb}_0(0), \mathsf{Hyb}_0(1)) \\
\leq & q^M \cdot \mathsf{TD}(\mathsf{Hyb}_1(0), \mathsf{Hyb}_1(1)) \qquad \text{(Lemma 57)} \\
\leq & q^M \cdot \mathsf{TD}(\mathsf{Hyb}_2(0), \mathsf{Hyb}_2(1)) \qquad \text{(Lemma 58)} \\
\leq & q^M \cdot (\mathsf{TD}(\mathsf{Hyb}_3(0), \mathsf{Hyb}_3(1)) + 2^{-\Omega(m)}) \qquad \text{(Lemma 59)} \\
\leq & q^M \cdot 2^{-\Omega(m)} \qquad \text{(Lemma 60)} \\
= & \mathrm{negl}(\lambda).
\end{aligned} \tag{36}$$

□

Lemma 57. $\mathsf{TD}(\mathsf{Hyb}_1(0), \mathsf{Hyb}_1(1)) \geq \mathsf{TD}(\mathsf{Hyb}_0(0), \mathsf{Hyb}_0(1))/q^M$.

Proof. We have $\mathsf{Hyb}_1(b)$ is identical to $\mathsf{Hyb}_0(b)$ if and only if $r_i = \mathsf{msg}_i^{(b)} + \sum_{j:\theta_j=0} x_i[j]$ for every $i \in [M]$, which happens with probability $1/q^M$. □

Lemma 58. $\mathsf{TD}(\mathsf{Hyb}_1(b), \mathsf{Hyb}_2(b)) = 0$.

Proof. The measurements on $\{\mathsf{C}_i\}_{i\in[M]}$ commute with all operations performed by $\mathcal{A}$ on its registers before the deletion certificates are returned. Moreover, $\boldsymbol{\theta}$ is sampled independently of all other variables. Hence, postponing the choice of $\boldsymbol{\theta}$ and the corresponding measurements of $\{\mathsf{C}_i\}_{i\in[M]}$ does not change the joint distribution of the adversary's residual state, its deletion certificates, and the eventual measurement outcomes. Therefore, $\mathsf{TD}(\mathsf{Hyb}_1(b), \mathsf{Hyb}_2(b)) = 0$. □

Lemma 59. $\mathsf{TD}(\mathsf{Hyb}_2(b), \mathsf{Hyb}_3(b)) \leq 2^{-\Omega(m)}$.

Proof. We first bound the probability that, for some $i \in \mathcal{S}$, the deletion certificate would be accepted in $\mathsf{Hyb}_2(b)$ but the additional measurement in $\mathsf{Hyb}_3(b)$ rejects. Let $\mathbf{y}_i$ denote the outcome of conceptually measuring C_i in the inverse-QFT basis. It suffices to bound

$$\Pr_{\boldsymbol{\theta}, \{\mathbf{y}_i\}_{i\in\mathcal{S}}} \left[\exists i \in \mathcal{S} : \mathsf{HW}\big(\mathbf{y}_i[\boldsymbol{\theta}_0] - \mathsf{cert}_i[\boldsymbol{\theta}_0]\big) \geq (m'-1)/2 \ \wedge \ \mathbf{y}_i[\boldsymbol{\theta}_1] = \mathsf{cert}_i[\boldsymbol{\theta}_1] \right].$$

By the union bound, we only need to prove for every $i \in \mathcal{S}$,

$$\Pr_{\boldsymbol{\theta},\mathbf{y}_i}[\mathsf{HW}(\mathbf{y}_i[\boldsymbol{\theta}_0] - \mathsf{cert}_i[\boldsymbol{\theta}_0]) \geq (m'-1)/2 \wedge \mathbf{y}_i[\boldsymbol{\theta}_1] = \mathsf{cert}_i[\boldsymbol{\theta}_1]] = 2^{-\Omega(m)}.$$

Define the event $\mathsf{Good} := \{m' \geq m/3\}$. By a Chernoff bound, $\Pr[\neg\mathsf{Good}] \leq \exp(-\Omega(m))$. Since $\mathsf{HW}(\mathbf{y}_i[\boldsymbol{\theta}_0] - \mathsf{cert}_i[\boldsymbol{\theta}_0]) \leq \mathsf{HW}(\mathbf{y}_i - \mathsf{cert}_i)$, we have

$$\begin{aligned}
&\Pr_{\boldsymbol{\theta},\mathbf{y}_i}[\mathsf{HW}(\mathbf{y}_i[\boldsymbol{\theta}_0] - \mathsf{cert}_i[\boldsymbol{\theta}_0]) \geq (m'-1)/2 \wedge \mathbf{y}_i[\boldsymbol{\theta}_1] = \mathsf{cert}_i[\boldsymbol{\theta}_1]] \\
\leq\ & \Pr[\neg\mathsf{Good}] + \Pr_{\boldsymbol{\theta},\mathbf{y}_i}[\mathsf{HW}(\mathbf{y}_i[\boldsymbol{\theta}_0] - \mathsf{cert}_i[\boldsymbol{\theta}_0]) \geq (m'-1)/2 \wedge \mathbf{y}_i[\boldsymbol{\theta}_1] = \mathsf{cert}_i[\boldsymbol{\theta}_1] \wedge \mathsf{Good}] \\
\leq\ & \Pr[\neg\mathsf{Good}] + \Pr[\mathsf{HW}(\mathbf{y}_i - \mathsf{cert}_i) \geq m/7 \wedge \mathbf{y}_i[\boldsymbol{\theta}_1] = \mathsf{cert}_i[\boldsymbol{\theta}_1]] \\
\leq\ & 2^{-\Omega(m)} + 2^{-m/7} = 2^{-\Omega(m)}.
\end{aligned}$$

Indeed, by the deferred generation of $\boldsymbol{\theta}$ in $\mathsf{Hyb}_2(b)$, conditioned on any fixed $\mathbf{y}_i$ and cert_i, the basis string $\boldsymbol{\theta}$ remains uniformly distributed. Hence, if $\mathsf{HW}(\mathbf{y}_i - \mathsf{cert}_i) \geq m/7$, the equality $\mathbf{y}_i[\boldsymbol{\theta}_1] = \mathsf{cert}_i[\boldsymbol{\theta}_1]$ requires $\theta[j] = 0$ for every j such that $y_i[j] \neq \mathsf{cert}_i[j]$, which occurs with probability at most $2^{-m/7}$. Since $|\mathcal{S}| \leq M = \mathrm{poly}(\lambda)$, the union bound over all $i \in \mathcal{S}$ preserves negligibility.

It remains to account for the disturbance caused by the additional measurements. If a deletion certificate would be rejected in $\mathsf{Hyb}_2(b)$, both experiments output $\perp$, so this branch contributes no distinguishing advantage. On the branch where the certificates would be accepted, the probability mass removed by the projections $\{\boldsymbol{\Pi}_{\mathsf{cert}_i}\}_{i\in\mathcal{S}}$ is $2^{-\Omega(m)}$ by the above argument. Therefore, by the Gentle Measurement Lemma (Lemma 4), performing these additional measurements changes the resulting state by at most $2^{-\Omega(m)}$. Hence, $\mathsf{TD}(\mathsf{Hyb}_2(b), \mathsf{Hyb}_3(b)) \leq 2^{-\Omega(m)}$. □

Lemma 60. $\mathsf{TD}(\mathsf{Hyb}_3(0), \mathsf{Hyb}_3(1)) = 0$.

Proof. Let $\mathcal{S} = \{s_1, \ldots, s_t\}$, where $t = |\mathcal{S}|$, and define a sequence of hybrids $\mathsf{G}_0, \ldots, \mathsf{G}_t$. For every $k \in \{0, \ldots, t\}$, G_k is defined in the same way as $\mathsf{Hyb}_3(0)$, except that the challenge message at index s_ℓ is changed from $\mathsf{msg}_{s_\ell}^{(0)}$ to $\mathsf{msg}_{s_\ell}^{(1)}$ for every $\ell \in [k]$. Since $\mathsf{msg}_i^{(0)} = \mathsf{msg}_i^{(1)}$ for every $i \in [M] \setminus \mathcal{S}$, we have $\mathsf{G}_0 = \mathsf{Hyb}_3(0)$ and $\mathsf{G}_t = \mathsf{Hyb}_3(1)$.

We prove that $\mathsf{TD}(\mathsf{G}_{k-1}, \mathsf{G}_k) = 0$ for every $k \in [t]$. Fix any $k \in [t]$ and let $i := s_k$. Denote $p_i := \sum_{j:\theta[j]=0} x_i[j]$, and let E_i contain all registers other than $C_i[\theta_0]$, including $C_i[\theta_1]$, the adversary's residual register, the secret key sk, all classical ciphertexts and deletion certificates, as well as the independently sampled value r_i. Thus, all possible correlations between the i-th ciphertext and the remaining system, including the post-deletion exposure of sk, are included in E_i.

Conditioned on the additional measurement accepting, the register $\mathsf{C}_i[\boldsymbol{\theta}_0]$ is supported on inverse-QFT basis states whose Hamming distance from $\mathsf{cert}_i[\boldsymbol{\theta}_0]$ is less than $(m'-1)/2$. By Lemma 5[10], p_i is uniformly distributed over $\mathbb{Z}_q$ and independent of E_i. In particular, conditioned on any fixed E_i (and hence on r_i), the condition $r_i = \mathsf{msg}_i^{(b)} + p_i$ holds with probability exactly $1/q$, for either $b \in \{0, 1\}$. Therefore, conditioned on the additional measurement accepting, replacing $\mathsf{msg}_i^{(0)}$ by $\mathsf{msg}_i^{(1)}$ does not change the distribution of the remaining system.

[10]To capture the notation in Lemma 5, we let $\mathsf{A} := \mathsf{E}_i$, $\mathsf{X} := \mathsf{C}_i[\boldsymbol{\theta}_0]$, and $\mathbf{R} := (1, \ldots, 1) \in \mathbb{Z}_q^{1\times m'}$. Since every column of $\mathbf{R}$ is nonzero, every set of one column of $\mathbf{R}$ is linearly independent, as required by the lemma. After applying the QFT transform, coherently computing $\mathbf{R}\mathbf{x}_i[\boldsymbol{\theta}_0] = p_i$, and tracing out $\mathsf{C}_i[\boldsymbol{\theta}_0]$, the resulting value p_i is uniformly distributed over $\mathbb{Z}_q$ and independent of E_i.

Before the final consistency check $r_i = \mathsf{msg}_i^{(b)} + p_i$, the distributions of G_{k-1} and G_k are identical. Hence the rejection branch of the additional measurement is identical in the two experiments. Conditioned on the additional measurement accepting, the argument above shows that the final consistency check also produces the same distribution in the two experiments. Therefore, $\mathsf{TD}(\mathsf{G}_{k-1}, \mathsf{G}_k) = 0$.

It follows that $\mathsf{TD}(\mathsf{Hyb}_3(0), \mathsf{Hyb}_3(1)) = \mathsf{TD}(\mathsf{G}_0, \mathsf{G}_t) = 0$. Since sk is included in E_i, the above equality holds jointly with the post-deletion exposure of the secret key. This finishes the proof.

A.4 Extension to BB84-Based Public-Key Compilers

Beyond the linearly homomorphic symmetric-key construction discussed above, our technique also applies more broadly to the BB84-based compiler paradigm of Bartusek and Khurana [BK23]. In particular, their framework compiles a variety of encryption primitives into corresponding notions with certified deletion, including public-key encryption, attribute-based encryption, and related variants. Our linear-homomorphic structure theorem suggests that, whenever the underlying encryption primitive admits a linear homomorphism, the corresponding BB84-based compiler can also preserve such a homomorphic structure, provided that the ciphertexts involved in the evaluation share the same basis string $\boldsymbol{\theta}$. A subtlety arises, however, in the public-key setting: in their compiler (see, e.g., [BK23, Corollary 4.8]), the basis string $\boldsymbol{\theta}$ is sampled freshly during each encryption call. Consequently, ciphertexts produced by independent encryption calls generally use different basis strings and cannot be directly combined using our transformation. Nevertheless, if the compiler is instantiated with a batch- or vector-encryption interface in which multiple ciphertext coordinates are generated using a common freshly sampled $\boldsymbol{\theta}$, then our technique yields a one-time linear homomorphism among those coordinates. A formal treatment of this batch variant is beyond the scope of this work.

It is also worth highlighting the case of fully homomorphic encryption with certified deletion. The FHE-with-certified-deletion construction of Bartusek and Khurana [BK23] supports deletion only after the computation is coherently reversed back to the original ciphertext; that is, the server eventually deletes the original input ciphertext rather than an already evaluated ciphertext. By contrast, in our construction, the permitted linear-homomorphic update preserves the BB84 form of the ciphertext together with a correspondingly updated verification key. Consequently, an evaluated ciphertext remains directly deletable and its deletion certificate remains verifiable, without coherently reversing the evaluation to recover the original input ciphertext. This structural property is not provided by their FHE-with-certified-deletion construction.