Communication-Efficient Threshold-CKKS Aggregation for Secure Cross-Silo Federated Learning

Jihyung Kook[1], Dokyeong Kang[1], and Eunsang Lee[1]

Department of Software, Sejong University, Seoul, Republic of Korea
jhkook30@gmail.com; s123kdg@gmail.com; eslee3209@sejong.ac.kr

Abstract. Federated learning (FL) enables collaborative training without sharing raw data, but exchanged model updates may reveal sensitive information. Homomorphic encryption (HE) protects these updates, yet combining collusion resistance, flexible aggregation, dropout handling, and communication efficiency remains challenging. We propose a communication-efficient threshold-CKKS protocol for cross-silo FL with Full and Sparse profiles, without a trusted key-distribution center. Independent local encryption keys enable server-side weighting and dropout-aware reaggregation without ciphertext retransmission. Sparse uses RLWE–MLWE conversion for fine-grained chunk-wise aggregation while retaining compact RLWE packing. Seeded ciphertext reconstruction, selective coefficient transmission, and compressed collective decryption further reduce communication. We establish whole-view simulation security for fixed-cohort executions without dropout or retry against a semi-honest server colluding with up to c clients, under explicitly permitted leakage and a minimum cohort requirement of $\tau \geq c + 2$. Experiments on CIFAR-10 and CIFAR-100 show that Full incurs only 0.32 and 0.29 percentage-point accuracy losses, with 1.78× and 1.76× the online communication of plaintext, respectively. At matched accuracy levels, Sparse reduces online communication by factors of approximately 79 and 84 compared with FedPHE. At accuracy targets five percentage points below full-transmission plaintext, Sparse requires only 2.5× and 2.1× the online communication of plaintext, respectively. These results narrow the communication gap between encrypted and plaintext FL.

Keywords: Federated learning · Homomorphic encryption · Threshold CKKS · Secure aggregation · Collective decryption · Model sparsification

1 Introduction

Federated learning (FL) enables multiple clients to collaboratively train a global model without centralizing their raw training data. Nevertheless, the model updates exchanged during training may themselves reveal sensitive information, motivating cryptographic mechanisms for protecting client updates during aggregation. One prominent line of work is masking-based secure aggregation, which

conceals individual client updates while allowing the server to recover an aggregate such as their sum or average [2, 3, 13]. Another line employs homomorphic encryption (HE), under which client updates remain encrypted while the server performs aggregation directly in the ciphertext domain [12, 16]. The latter provides a natural basis for supporting aggregation functionality beyond a protected sum or average.

Early additively homomorphic FL approaches based on a client-shared decryption key protect client updates from the server alone, but lose this protection if the server colludes with a key-holding client [12]. FedPHE represents an important step toward richer encrypted aggregation by supporting server-side weighting while explicitly considering server–client collusion [7]. Its design, however, relies on a trusted key-distribution center and scalar obfuscation. Because the same scalar mask is applied within a pack, pairwise differences between packed values are preserved, so the masking itself does not hide all intra-pack relational information when an obfuscated pack is available to the colluding parties. Moreover, FedPHE does not provide a survivor-specific re-aggregation mechanism for clients that drop out after uploading their local ciphertexts.

A natural way to strengthen collusion resistance without giving any client complete decryption capability is to distribute decryption across clients using multi-key or threshold homomorphic encryption. xMK-CKKS is a representative example of this approach: client updates are encrypted under an aggregated public key, and decryption requires collaboration among the participating clients [9]. Such collaborative decryption is particularly relevant to cross-silo FL, where the number of participating organizations is moderate and participation is relatively stable. However, xMK-CKKS essentially supports the summation of encrypted client updates and does not address server-side post-encryption weighting, client dropout, or model-update sparsification. Moreover, its multi-key CKKS construction and collaborative decryption incur substantial communication overhead. These limitations leave open the challenge of simultaneously supporting flexible encrypted aggregation, collusion resistance, dropout handling, sparsification, and communication efficiency in a practical cross-silo FL protocol.

To address these limitations, we develop an integrated threshold-CKKS-based federated learning protocol for cross-silo FL that supports server-side post-encryption weighting while jointly addressing server–client collusion, minimum cohort requirements, client dropout, and model-update sparsification, without relying on a trusted key-distribution center. An overview of the proposed protocol is shown in Fig. 1. We further incorporate communication-reduction techniques throughout encrypted update transmission, aggregation, and distributed decryption, and provide a formal security analysis under the explicitly stated threat model, cryptographic assumptions, and execution scope.

1.1 Contributions

– **Collusion-aware and dropout-resilient threshold aggregation.** We develop a threshold-CKKS-based FL protocol that addresses client dropouts

and collusion between a semi-honest server and up to c clients, without relying on a trusted key-distribution center. The protocol enforces a minimum cohort of $\tau \geq c+2$, including label-wise cohort control in the Sparse profile, and handles dropouts without retransmitting previously uploaded local ciphertexts. It also supports server-side application of setup-fixed aggregation weights to encrypted updates, without requiring client-side pre-weighting.
- **Fine-grained sparse aggregation.** We develop an RLWE–MLWE-conversion-based aggregation mechanism that enables fine-grained sparsification while preserving compact RLWE packing. By converting packed RLWE ciphertexts into an MLWE representation, the server can independently align, weight, and aggregate client-specific sparse chunks. The aggregated chunks are then repacked into RLWE ciphertexts for collective decryption. This design decouples sparsification granularity from the RLWE packing dimension, enabling finer-grained selection of important updates and reducing accuracy loss under a given communication budget.
- **Communication-efficient realization.** We reduce communication across the entire threshold-CKKS aggregation pipeline. Seeded reconstruction of uniform ciphertext components and selective transmission of sparse coefficients reduce uplink traffic. Compression of uploaded ciphertexts and aggregate-a components, together with code-sum compression of partial-decryption responses, further reduces communication during collective aggregation and decryption.
- **Formal security and empirical validation.** We establish whole-view simulation security for both Full and Sparse under fixed-cohort executions, accounting for the compressed messages observed by a colluding server and clients. We implement the protocol using PyTorch and OpenFHE and evaluate it on CIFAR-10 and CIFAR-100. Full achieves nearly the same accuracy as plaintext, with losses of only 0.32 and 0.29 percentage points, while requiring only 1.78× and 1.76× the online communication, respectively. At matched accuracy levels, Sparse reduces online communication by approximately 79× and 84× compared with FedPHE. At accuracy targets five percentage points below full-transmission plaintext, Sparse incurs only 2.5× and 2.1× the online communication of plaintext, respectively.

1.2 Related Work

Server-side post-encryption weighting. Masking-based secure aggregation protects client updates through masks that cancel during summation [2, 3, 13], including variants supporting sparsification [8]. However, these mechanisms do not directly support client-specific weights assigned after protected updates have been uploaded. SWAS [5] avoids requiring clients to know their normalized aggregation weights, but requires each client to pre-scale its update by its local dataset size. In contrast, our clients encrypt unweighted updates, allowing the server to apply setup-fixed weights in the ciphertext domain without requiring client-side pre-scaling.

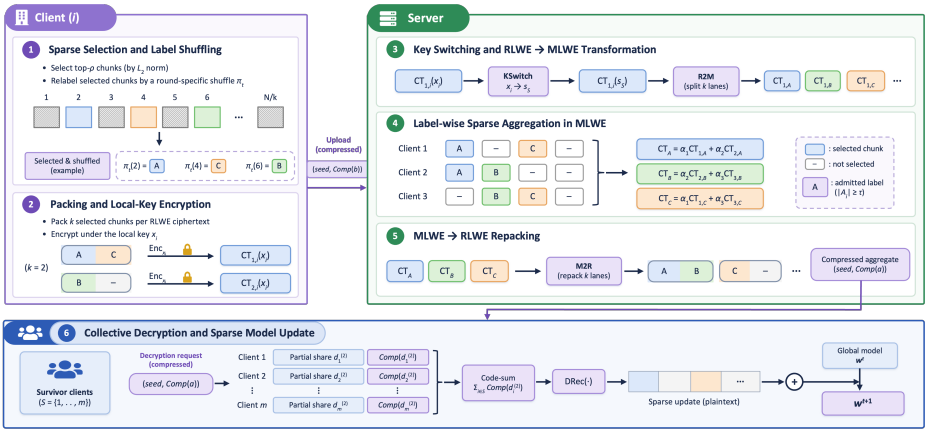


Fig. 1. Sparse-profile workflow for $k > 1$: chunk selection and label shuffling, local-key encryption, key switching, RLWE–MLWE conversion and label-wise aggregation, MLWE–RLWE repacking, and compressed collective decryption. The survivor-client stage also illustrates the operational dropout extension; the whole-view security theorem concerns fixed-cohort runs without dropout or retry.

Collusion-resistant HE aggregation. HE-based FL schemes using a client-shared decryption key are vulnerable to server–client collusion [12, 16]. FedPHE [7] addresses collusion through scalar obfuscation but relies on a trusted key-distribution center. Moreover, its common scalar mask preserves intra-pack pairwise differences when an obfuscated pack is exposed to colluding parties. xMK-CKKS [9] distributes decryption across clients, but does not address dropout or sparsification and incurs substantial communication overhead. Its security analysis does not establish simulation-based security for the joint server–colluder view.

Sparse and communication-efficient HE aggregation. FedPHE performs Top-K selection at the CKKS-pack level, tying sparsification granularity to the packing dimension. FedSHE-CQ [15] combines Top-K sparsification with compressed indices and quantization, but uses a union representation for client-dependent supports, potentially allocating ciphertext positions to unselected coordinates. Our RLWE–MLWE conversion enables independent chunk-level alignment and weighted aggregation while retaining compact RLWE packing. Seeded uploads, selective b-coefficient transmission, and collective-decryption compression further reduce communication.

2 Preliminaries

2.1 Federated Learning and Update Sparsification

In synchronous federated learning, clients train a common model while keeping their training data local. In FedAvg-style training, client i starts round t from

$w^{(t)} \in \mathbb{R}^d$, performs E local epochs, and obtains $w_i^{(t,E)}$. Let $\delta_i^{(t)} = w_i^{(t,E)} - w^{(t)}$ denote the local update. For a full-participation round with n clients, model averaging is

$$w^{(t+1)} = w^{(t)} + \sum_{i=1}^{n} \beta_i \delta_i^{(t)},$$

where $\beta_i \geq 0$ are real-valued weights with $\sum_i \beta_i = 1$, usually proportional to local dataset sizes [10]. This equation describes the dense averaging reference, before sparsification or cryptographic protection.

Update sparsification reduces communication by sending selected coordinates and their positions rather than the entire vector. Chunk-level sparsification groups coordinates into fixed-length blocks and selects blocks by an importance measure, such as their Euclidean norm. Selection can differ across clients, so their transmitted positions must be aligned before aggregation. Error feedback retains omitted components in a local residual and adds them to a later update before selection [14]. Thus a dense update carries all coordinates, whereas a sparsified update carries only selected coordinates or chunks.

2.2 CKKS Encoding and Key Switching

For ciphertext modulus Q and power-of-two degree N, let $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$. We also use $R_{QP} = \mathbb{Z}_{QP}[X]/(X^N + 1)$, where P is the special modulus for key switching. Write $Q = \prod_{\lambda=0}^{m-1} q_\lambda$ with pairwise coprime odd RNS primes q_λ and P coprime to Q. Define the CRT gadget polynomials' scalar coefficients by

$$\widehat{q}_\lambda = Q/q_\lambda, \qquad \gamma_\lambda = (\widehat{q}_\lambda^{-1} \bmod q_\lambda)\widehat{q}_\lambda.$$

Then $\gamma_\lambda \equiv 1 \pmod{q_\lambda}$ and $\gamma_\lambda \equiv 0 \pmod{q_\mu}$ for $\mu \neq \lambda$. The proof-certified parameter family uses $m = 2$, while the subscript notation permits any fixed RNS count. Write $f[j]$ for coefficient j, $[x]_Q$ for the residue in $\{0, \ldots, Q-1\}$, $\mathsf{Center}_Q(x)$ for its centered representative, and $\lfloor x \rceil$ for nearest-integer rounding. These conventions apply coefficientwise.

CKKS is a homomorphic encryption scheme based on ring learning with errors (RLWE) that supports approximate arithmetic on encrypted real and complex numbers. Its canonical encoding packs values into slots for parallel, single-instruction multiple-data (SIMD) operations [4]. We instead use the underlying RLWE ciphertext arithmetic with *direct coefficient encoding*: a real block $v \in \mathbb{R}^N$ is represented by $\mu[j] = \lfloor \Delta v[j] \rceil \bmod Q$ at scale $\Delta > 0$, with shorter blocks zero-padded. Thus each value occupies one polynomial coefficient rather than a canonical slot. This representation preserves coordinatewise addition and integer-scalar multiplication; we do not require slotwise ciphertext multiplication.

We order an RLWE ciphertext as $ct = (b, a) \in R_Q^2$, with $b + as = \mu + e \pmod{Q}$, where $s, \mu, e \in R_Q$ are the secret, plaintext, and encryption error. Decoding takes the centered coefficients and divides by Δ. We write $D_{\mathbb{Z},\sigma}$ for the centered integer Gaussian with mass proportional to $\exp(-x^2/(2\sigma^2))$.

Key switching transforms a ciphertext (b, a) under secret s into (b', a') under s' such that $b' + a's' = b + as + \varepsilon_{\mathrm{KS}} \pmod{Q}$, where $\varepsilon_{\mathrm{KS}}$ denotes the switching error.

2.3 Collective Decryption

Collective decryption distributes decryption capability among key holders rather than giving one party the complete secret [11]. For a fixed participating group $\mathcal{S}$, let each client hold an independent secret $s_i \in R_Q$, and let $s_{\mathcal{S}} = \sum_{i \in \mathcal{S}} s_i$. A ciphertext under this collective secret satisfies $b + as_{\mathcal{S}} = \mu + e \pmod{Q}$.

Each client i generates a *raw partial-decryption response* $h_i = as_i + g_i \pmod{Q}$ using its secret share s_i and fresh private smudging noise $g_i \in R_Q$, and sends h_i to the server. The server combines the responses with the ciphertext's b-component:

$$b + \sum_{i \in \mathcal{S}} h_i = \mu + e + \sum_{i \in \mathcal{S}} g_i \pmod{Q}.$$

It then decodes the aggregate without reconstructing $s_{\mathcal{S}}$. The added noise protects the secret-dependent responses while retaining decoding precision. This additive construction requires every member of $\mathcal{S}$, not an arbitrary subset of key holders.

2.4 RLWE–MLWE Conversion

We use the RLWE–MLWE conversion technique of HERMES [1] with an explicit coefficient convention. Assume $k \mid N$, put $B = N/k$, and work in $R_{Q,B} = \mathbb{Z}_Q[Y]/(Y^B + 1)$ with the embedding $Y = X^k$. A rank-k MLWE ciphertext $ct^{\mathrm{M}} = (c_0, \ldots, c_k) \in R_{Q,B}^{k+1}$ under $\mathbf{s}^{\mathrm{M}} = (s_0^{\mathrm{M}}, \ldots, s_{k-1}^{\mathrm{M}})$ obeys

$$c_k + \sum_{j=0}^{k-1} c_j s_j^{\mathrm{M}} = \mu^{\mathrm{M}} + e^{\mathrm{M}} \pmod{Q}.$$

Split, embed, and combine. For $f \in R_Q$, let

$$\mathsf{Split}_h(f)[x] = f[kx + h], \qquad 0 \le h < k, \quad 0 \le x < B.$$

For $g \in R_{Q,B}$ define $\mathsf{Embed}(g) = g(X^k)$, and for $g_0, \ldots, g_{k-1} \in R_{Q,B}$ set

$$\mathsf{Combine}(g_0, \ldots, g_{k-1}) = \sum_{h=0}^{k-1} X^h \mathsf{Embed}(g_h). \tag{1}$$

The maps are mutually inverse in the following sense:

$$\mathsf{Split}_h(\mathsf{Combine}(\mathbf{g})) = g_h,$$
$$\mathsf{Combine}((\mathsf{Split}_h(f))_{h<k}) = f,$$

where $\mathbf{g} = (g_0, \ldots, g_{k-1})$. For a ring secret s, write $\mathbf{s}^{\mathrm{M}} = (\mathsf{Split}_j(s))_{0 \le j < k}$ and $\Psi_j(s) = \mathsf{Embed}(\mathsf{Split}_j(s))$.

Exact ring-to-module conversion (R2M). Let $ct = (b, a)$ satisfy $b + as = \mu + e \pmod{Q}$. Put $\bar{b}_h = \mathsf{Split}_h(b)$ and $\bar{a}_r = \mathsf{Split}_r(a)$. For each $h < k$, define $ct_h^{\mathrm{M}} = (c_{h,0}, \ldots, c_{h,k})$ by

$$\begin{aligned} c_{h,k} &= \bar{b}_h, \\ c_{h,j} &= Y^{\mathbf{1}_{\{h<j\}}} \bar{a}_{[h-j]_k}, \qquad 0 \le j < k, \end{aligned} \tag{2}$$

where $[h-j]_k \in \{0, \ldots, k-1\}$ is the residue modulo k. The negacyclic indexing gives the exact identity

$$c_{h,k} + \sum_{j=0}^{k-1} c_{h,j} \mathsf{Split}_j(s) = \mathsf{Split}_h(\mu + e) \pmod{Q}. \tag{3}$$

Thus $\mathsf{R2M}_k(ct) = (ct_h^{\mathrm{M}})_{h<k}$ produces k MLWE encryptions of the individual coefficient lanes without an evaluation key or additional arithmetic error. In particular, if an untransmitted coefficient of b is replaced by zero, only its corresponding output lane changes.

Module-to-ring repacking (M2R). Suppose up to k MLWE ciphertexts $ct_h^{\mathrm{M}} = (c_{h,0}, \ldots, c_{h,k})$ share the secret vector $\mathbf{s}^{\mathrm{M}}$; pad unused ciphertexts by the all-zero MLWE tuple. Combine each column:

$$C_j = \mathsf{Combine}(c_{0,j}, \ldots, c_{k-1,j}), \qquad 0 \le j \le k.$$

The combined components satisfy

$$C_k + \sum_{j=0}^{k-1} C_j \Psi_j(s) = \mathsf{Combine}(\mu_0^{\mathrm{M}}, \ldots, \mu_{k-1}^{\mathrm{M}}) + E_{\mathrm{pre}} \pmod{Q}, \tag{4}$$

where E_{pre} is the similarly combined pre-switching MLWE error. Switching the k source terms from the secrets $\Psi_j(s)$ to a common target ring secret yields one RLWE ciphertext carrying the repacked plaintext. Unlike R2M, this switching step introduces additional error. Section 4.4 gives the exact switching-key construction and execution for the proposed protocol. The algebraic correctness of the conversion identities is proved in Section 5.2.

3 System and Security Model

3.1 System Model

We consider synchronous cross-silo FL with a server and registered clients $\mathcal{C} = \{1, \ldots, n\}$. We target environments with relatively stable participation, where client dropouts are expected to be occasional. Starting from the common model $w^{(t)} \in \mathbb{R}^d$, clients obtain local model changes $\delta_i^{(t)}$ as in Section 2.1. Clients encrypt their local model updates, and the server applies setup-fixed aggregation weights α_i in the ciphertext domain.

Each setup fixes one model-update profile $\mathsf{mode} \in \{\mathsf{SP}, \mathsf{FULL}\}$. In *Full*, each client contributes its complete model update for encrypted aggregation. In *Sparse*, each client partitions its update into length-B chunks, selects high-importance chunks according to the setup-fixed top-ρ rule, and contributes only the admitted selected chunks to the encrypted aggregation. In Sparse, clients refer to selected chunks by attempt-specific *shuffled labels* rather than actual chunk indices, allowing the server to match corresponding chunks across clients without directly using their model positions. In both profiles, changes to shared auxiliary states $z^{(t)} \in \mathbb{R}^{d_{\mathrm{aux}}}$, such as running statistics, are aggregated through a separate dense path without sparsification. Changing the profile, registered membership, or setup-fixed parameters requires a fresh setup; temporary dropout does not change registration.

A training round may consist of multiple attempts, each identified by a distinct rid_t. Only a successfully finalized attempt commits the model and auxiliary states and advances the round index from t to $t+1$. An aborted attempt leaves the previously committed states unchanged, and a retry uses a fresh attempt identifier. Before collective decryption begins, however, client dropout may be handled within the same attempt by reducing the survivor set. Within an attempt, $\mathcal{S}_t \subseteq \mathcal{C}$ denotes the current survivor set, which may shrink before collective decryption, and $\mathcal{S}_t^\star \subseteq \mathcal{S}_t$ denotes the final decryption cohort fixed before collective decryption begins.

3.2 Threat Model and Leakage

Parties are semi-honest. A classical probabilistic polynomial-time server may collude with at most c clients, with the colluding set fixed before setup. Colluders share their local inputs, secret shares, randomness, and received messages. We require $n \geq \tau \geq c+2$. Each released Sparse chunk must have at least τ contributors, while Full and auxiliary outputs require at least τ final survivors. Thus, after colluders subtract their own contributions, every released aggregate still contains contributions from at least two honest clients.

In Sparse, the server observes which clients selected each shuffled label and which labels were admitted, but when $c = 0$ it does not learn the mapping from shuffled labels to actual model-chunk indices. This guarantee does not extend to $c \geq 1$, since a colluding client knows the shared permutation key; shuffling does not hide support overlap or inference from aggregate values.

We treat the aggregate information required by the protocol and observable execution metadata as permitted leakage. Beyond this information, the joint server–colluder view should reveal no additional information about honest clients' individual updates or secret keys. The following subsection defines the allowed leakage and security goal for fixed-cohort executions. Operational dropout handling is described separately in Section 4.7.

3.3 Security Goal and Allowed Leakage

We establish whole-view simulation security for fixed-cohort executions without dropout or retry. The statement concerns the entire joint server–colluder view, conditional on explicitly permitted leakage.

Real view. Let $\mathcal{A} \subseteq \mathcal{C}$ denote the statically corrupted clients, with $|\mathcal{A}| \leq c$. The joint view of the semi-honest server and these clients over the entire execution is defined as

$$\mathsf{View}_{\mathcal{A}}^{\mathrm{real}} = (\mathsf{St}_S, \mathsf{Rnd}_S, \mathsf{Transcript}_{\mathrm{wire}}, \mathsf{CorSt}_{\mathcal{A}}), \tag{5}$$

where St_S includes the server's public and cached key material (e.g., p_1, $p_{0,\mathcal{C}}$, KSK rows $(B_{i,\lambda}, A_{i,\lambda})$, and M2R key shares and aggregates), reconstructed ciphertexts, and intermediate aggregation states. Rnd_S contains the server's protocol randomness. $\mathsf{Transcript}_{\mathrm{wire}}$ includes all observable messages, such as label bitmaps Y_i, Y_t, upload seeds $\zeta_{i,r}$, compressed upload codes $C_{i,r}^{(1)}$, decryption-request codes $\bar{a}_\nu$, and partial-decryption codes $C_{i,\nu}^{(2)}$. Finally, $\mathsf{CorSt}_{\mathcal{A}}$ contains the corrupted clients' secret keys, local inputs, private randomness, and application states. The view covers both model and auxiliary paths across all analyzed rounds.

Allowed leakage. We allow the simulator to access the public setup configuration Cfg, fixed aggregation weights $\boldsymbol{\alpha}$, public semantic execution trace Φ^{sem}, corrupted-client state $\mathsf{CorSt}_{\mathcal{A}}$, and aggregate values (M, O). In particular, Φ^{sem} includes the client cohort, execution identifiers, profile and path information, and, for Sparse, the shuffled-label bitmaps, admitted supports, upload supports, and final grouping and packing-lane assignments.

Let $M = (M^{\mathrm{model}}, M^{\mathrm{aux}})$ denote the exact encoded weighted aggregates, excluding cryptographic and compression errors, and let $O = (O^{\mathrm{model}}, O^{\mathrm{aux}})$ denote the raw outputs reconstructed by collective decryption, before final-delivery quantization. The simulator's permitted information is

$$\Lambda^{MO} = (\mathsf{Cfg}, \boldsymbol{\alpha}, \Phi^{\mathrm{sem}}, \mathsf{CorSt}_{\mathcal{A}}, M, O). \tag{6}$$

The exact semantics of this leakage and the conditioned transcript distribution are established in Sections 5 and 7. It does not include individual honest-client updates or secret keys.

Security goal. We aim to show that the entire server–colluder view can be simulated using only the allowed leakage Λ^{MO}. Formally, for every admissible static semi-honest adversary, there exists a uniform probabilistic polynomial-time simulator Sim such that

$$(\Lambda^{MO}, \mathsf{View}_{\mathcal{A}}^{\mathrm{real}}) \approx_c (\Lambda^{MO}, \mathsf{Sim}(\Lambda^{MO})), \tag{7}$$

where $\approx_c$ denotes computational indistinguishability. Furthermore, the simulated decryption transcript must reconstruct the prescribed raw output exactly:

$$\mathsf{Recover}(\mathsf{Sim}(\Lambda^{MO})) = O. \tag{8}$$

Here, Recover applies the code-sum reconstruction in (18) to all released coefficients, before final-delivery quantization. Complete security proofs for Full, Sparse with $k = 1$, and Sparse with $k > 1$ are presented in Sections 6–8.

3.4 Execution Scope and Randomness Model

The whole-view theorem covers a fixed registered cohort and a statically selected corrupted set, with no dropout, survivor change, or retry. Each round may select new Sparse supports based on previously released aggregate outputs and public application history; the theorem does not assume independent or fixed supports across rounds. However, the application is not allowed to adapt directly to hidden cryptographic transcript randomness rather than the prescribed semantic outputs. All clients in the fixed decryption cohort respond for every released Sparse label, even if they did not upload that label.

The proof works in the classical programmable ideal-XOF model and assumes exact sampling of the stipulated secret-key and quantizer randomness, or explicitly accounts for implementation-level distribution discrepancies. For Sparse, the shared permutation key is established through an ideal authenticated and confidential client-only channel whose implementation and channel transcript lie outside the proved server–colluder view. These assumptions are stated formally in the complete security proof. The theorem does not cover the separate operational dropout extension in Section 4.7.

4 Threshold-CKKS Federated Learning Protocol

4.1 Design Overview and Key Ideas

Our protocol supports two model-update profiles, *Full* and *Sparse*, built upon a common encrypted-aggregation and collective-decryption framework. We highlight five key design ideas below.

Independent key domains. Our protocol employs independent secrets for local encryption, collective aggregation, and repacked Sparse-model decryption (for $k > 1$). This separation establishes an acyclic key-switching structure, enabling an ssRLWE-based security proof without relying on circular-security or key-dependent-message (KDM) assumptions for evaluation keys.

Local encryption and survivor-key switching. Each client encrypts its updates under its own local secret key, and the server subsequently key-switches the uploaded ciphertexts to the collective key for aggregation. This enables seeded reconstruction of uniform a-components, reducing upload communication. It also permits survivor-specific reaggregation of already uploaded ciphertexts before decryption, without ciphertext retransmission.

Fine-grained sparse aggregation. We repurpose the RLWE–MLWE conversion technique of HERMES [1] for chunk-wise sparse aggregation in FL. Clients pack up to k selected chunks of length $B = N/k$ into a single RLWE ciphertext. The server uses ring-to-module (R2M) conversion to separate the chunks

for label-wise weighted aggregation, followed by module-to-ring (M2R) conversion to repack the aggregates for collective decryption. This enables fine-grained sparsification while retaining efficient RLWE packing.

Compression across the communication pipeline. We introduce a stage-specific compression scheme spanning client uploads, collective-decryption requests, partial-decryption responses, and final delivery, using bit widths (c_1, c_a, c_2, c_d). In particular, we compress aggregate-a components and employ code-sum compression for partial-decryption responses, extending communication savings to collective decryption. Stochastic rounding is used where needed to obtain conditionally centered errors for the security analysis. The Sparse profile further reduces upload traffic by transmitting only selected coefficients of the ciphertext b-component.

Dropout-aware reaggregation and collective decryption. Our protocol employs collective decryption without reconstructing the collective secret key. To accommodate client dropouts, the server can reconfigure the collective key and reaggregate previously uploaded ciphertexts for the surviving clients before decryption begins, without requiring ciphertext retransmission. Once decryption begins, the participant set is fixed; a missing response aborts the attempt without committing model, auxiliary, or error-feedback state updates.

4.2 Setup and Key Material

Secret key generation. Each client $i \in \mathcal{C}$ independently samples $x_i, s_i \leftarrow \chi_u$, where $\chi_u = U(\{-1, 0, 1\})^N$. For Sparse with $k > 1$, it additionally samples an independent $t_i \leftarrow \chi_u$. The secret x_i is used for local encryption, s_i for the collective key domain, and t_i for the repacked Sparse-model decryption domain. All secret keys remain local to client i.

Public uniform-polynomial reconstruction. For reproducible seed-derived ring masks, define $\mathsf{UPoly}_M(\mathsf{seed}, \mathsf{tag}) \in R_M$ for $M \in \{Q, QP\}$ as follows. For every prime $\mu \mid M$, use the domain-separated SHAKE128 stream

$$\mathsf{SHAKE128}(\texttt{UPOLY} \| \mathsf{seed} \| \mathsf{tag} \| \mathsf{BE64}(N) \| \mathsf{BE64}(\mu)),$$

where $\|$ denotes unambiguous byte concatenation, $\mathsf{BE64}$ is unsigned 64-bit big-endian encoding, and all ring primes satisfy $\mu < 2^{64}$. Read successive unsigned 64-bit words w, discarding a word unless $w < L_\mu := \lfloor 2^{64}/\mu \rfloor \mu$; an accepted word yields the next coefficient $w \bmod \mu$. Rejected words are consumed, and the independently generated prime residues are combined by CRT. This specifies an unbiased residue sampler from the XOF stream and a deterministic reconstruction from a fixed seed; a bounded sampler's exceptional failure is handled in the security and correctness conditions. The server samples fresh public setup and CRS identifiers and sets

$$p_1 = \mathsf{UPoly}_{QP}(\mathsf{seedCRS}, \texttt{PK-P1} \| \mathsf{setup_id}).$$

Local encryption uses a fresh 128-bit private seed per client, path, block, and attempt, with distinct `ENC-FULL-A`, `ENC-SP-A`, and `ENC-AUX-A` tags as appropriate.

Collective public key. At setup, the server broadcasts the CRS seed and setup identifier; all clients regenerate the uniform polynomial $p_1 \in R_{QP}$ using the preceding reconstruction rule. Each client $i \in \mathcal{C}$ samples $f_i \leftarrow \chi_e$ and uploads its public-key share $p_{0,i} = -p_1 s_i + f_i \pmod{QP}$. The server then computes and broadcasts $p_{0,\mathcal{C}} = \sum_{i\in\mathcal{C}} p_{0,i}$. The resulting collective public key $(p_{0,\mathcal{C}}, p_1)$ corresponds to $s_{\mathcal{C}} = \sum_{i\in\mathcal{C}} s_i$, with public-key noise $f_{\mathcal{C}} = \sum_{i\in\mathcal{C}} f_i$; the collective secret is never reconstructed. Importantly, this public key is used to generate client-to-collective switching keys, rather than to encrypt local updates directly.

Client-to-collective switching keys. Using the collective public key $(p_{0,\mathcal{C}}, p_1)$, each client i generates a HYBRID key-switching key $\mathrm{KSK}_{i\to\mathcal{C}} : x_i \to s_{\mathcal{C}}$. For each gadget component λ, the client independently samples $u_{i,\lambda} \leftarrow \chi_u$ and $e^0_{i,\lambda}, e^1_{i,\lambda} \leftarrow \chi_e$, and computes

$$\begin{aligned} A_{i,\lambda} &= u_{i,\lambda} p_1 + e^1_{i,\lambda}, \\ B_{i,\lambda} &= u_{i,\lambda} p_{0,\mathcal{C}} + e^0_{i,\lambda} + P\gamma_\lambda x_i \pmod{QP}, \end{aligned}$$

where γ_λ denotes the CRT gadget of Section 2.2. The switching-key row satisfies $B_{i,\lambda} + A_{i,\lambda} s_{\mathcal{C}} = P\gamma_\lambda x_i + e^{\mathrm{KS}}_{i,\lambda}$, with $e^{\mathrm{KS}}_{i,\lambda} = e^0_{i,\lambda} + e^1_{i,\lambda} s_{\mathcal{C}} + u_{i,\lambda} f_{\mathcal{C}}$ in R_{QP}. Each client uploads $\mathrm{KSK}_{i\to\mathcal{C}} = \{(B_{i,\lambda}, A_{i,\lambda})\}_{\lambda=0}^{m-1}$ to the server once during setup. The server caches the key for switching locally encrypted ciphertexts to the collective key domain across rounds. A survivor-specific target key is generated only as needed during the operational dropout extension.

Label-permutation and M2R keys. Sparse clients establish a shared secret permutation key $K_{\mathrm{perm}} \in \{0,1\}^{128}$ through an authenticated and confidential channel. This key, unavailable to the server, is used to generate common attempt-specific permutations of chunk labels.

For $k > 1$, recall $\Psi_j(s) = \mathsf{Embed}(\mathsf{Split}_j(s))$ from Section 2.4. The CRS seed determines the common uniform polynomials

$$\begin{aligned} a^M_{j,\lambda} = \mathsf{UPoly}_{QP}(&\mathsf{seedCRS}, \texttt{M2R-A}\|\mathsf{setup_id} \\ &\|\mathsf{BE64}(j)\|\mathsf{BE64}(\lambda)) \end{aligned}$$

for $0 \le j < k$ and $0 \le \lambda < m$. Each client i independently samples $e^M_{j,\lambda,i} \leftarrow \chi_e$ and computes

$$b^M_{j,\lambda,i} = -a^M_{j,\lambda} t_i + P\gamma_\lambda \Psi_j(s_i) + e^M_{j,\lambda,i} \pmod{QP}.$$

Each client uploads these M2R key shares to the server once during setup. The server caches them and constructs $B^M_{j,\lambda,\mathcal{C}} = \sum_{i\in\mathcal{C}} b^M_{j,\lambda,i}$, yielding the M2R switching key $\mathrm{swk}^{\mathrm{M2R}}_{j,\mathcal{C}} = \{(B^M_{j,\lambda,\mathcal{C}}, a^M_{j,\lambda})\}_{\lambda=0}^{m-1}$ from $\Psi_j(s_{\mathcal{C}})$ to $t_{\mathcal{C}} = \sum_{i\in\mathcal{C}} t_i$. Indeed, each row satisfies $B^M_{j,\lambda,\mathcal{C}} + a^M_{j,\lambda} t_{\mathcal{C}} = P\gamma_\lambda \Psi_j(s_{\mathcal{C}}) + \sum_{i\in\mathcal{C}} e^M_{j,\lambda,i} \pmod{QP}$. No M2R keys are required for $k = 1$.

Server-side weights. At setup, the server assigns each client $i \in \mathcal{C}$ a positive integer weight α_i approximating its dataset fraction β_i, such that $\alpha_i/128 \approx \beta_i$ and $\sum_{i\in\mathcal{C}} \alpha_i = 128$. These weights remain fixed throughout training, and the aggregated updates are divided by 128 after decryption. Neither missing Sparse contributors nor client dropouts trigger renormalization of the surviving weights.

4.3 HYBRID Key Switching: Full and Sparse

The switching-key rows constructed during setup are cached, but key-switching arithmetic is performed afresh for each source ciphertext. We specify both the deterministic Full routine and the randomized Sparse routine. The latter is used for Sparse model and auxiliary ciphertexts and for Sparse M2R repacking; Full uses the former.

Switching-key interface. For a source secret z and target secret z', suppose $\mathrm{KSK}_{z\to z'} = \{(B_\lambda, A_\lambda)\}_{\lambda<m}$ satisfies in R_{QP}

$$B_\lambda + A_\lambda z' = P\gamma_\lambda z + e_\lambda^{\mathrm{KS}} \pmod{QP}. \tag{9}$$

The client-to-collective key is the instance $z = x_i$, $z' = s_{\mathcal{C}}$; the M2R keys have $z = \Psi_j(s_{\mathcal{C}})$, $z' = t_{\mathcal{C}}$. For $(b, a) \in R_Q^2$, construct one decomposition polynomial $d_\lambda \in R_Q$ for each RNS prime and form

$$U_B = \sum_{\lambda=0}^{m-1} d_\lambda B_\lambda, \qquad U_A = \sum_{\lambda=0}^{m-1} d_\lambda A_\lambda \quad \text{in } R_{QP}.$$

The sums are evaluated in R_{QP} *before* either component is reduced from modulus QP to Q. Both components use the same decomposition digits.

Deterministic Full switching. For each coefficient u let $d_\lambda[u] = [a[u]]_{q_\lambda} \in \{0, \ldots, q_\lambda - 1\}$. For $U \in \{U_B, U_A\}$, let $\rho_U[u] = \mathsf{Center}_P([U[u]]_P)$ and set

$$\mathsf{ModDown}(U) = (U - \rho_U)/P \pmod{Q}.$$

The Full key switch outputs $(b', a') = (b + \mathsf{ModDown}(U_B), \mathsf{ModDown}(U_A))$. Its arithmetic remainder is deterministic once the source ciphertext and cached key are fixed.

Randomized signed-digit Sparse switching. For each λ, u compute $r_{\lambda,u} = [a[u]]_{q_\lambda}$ and draw fresh, independent $\omega_{\lambda,u}^{\mathrm{dig}} \leftarrow U(\mathbb{Z}_{q_\lambda})$. Set

$$d_\lambda^\circ[u] = r_{\lambda,u} - q_\lambda \mathbf{1}_{\{\omega_{\lambda,u}^{\mathrm{dig}} < r_{\lambda,u}\}}. \tag{10}$$

Then $d_\lambda^\circ \equiv a \pmod{q_\lambda}$, $\mathbb{E}[d_\lambda^\circ[u] \mid a] = 0$, and $\mathbb{E}[(d_\lambda^\circ[u])^2 \mid a] = r_{\lambda,u}(q_\lambda - r_{\lambda,u}) \le q_\lambda^2/4$. Use these same signed digits in U_B and U_A.

For each $U \in \{U_B, U_A\}$ and coefficient u, put $r_U[u] = [U[u]]_P$, sample fresh $\omega^{\mathrm{MD}}_{U,u} \leftarrow U(\mathbb{Z}_P)$, and define

$$\begin{aligned} \rho^{\circ}_U[u] &= r_U[u] - P\mathbf{1}_{\{\omega^{\mathrm{MD}}_{U,u} < r_U[u]\}}, \\ \mathsf{ModDown}^{\circ}(U) &= (U - \rho^{\circ}_U)/P \pmod{Q}. \end{aligned} \tag{11}$$

The B- and A-component ModDown coins are independent, with $\mathbb{E}[\rho^{\circ}_U[u] \mid U] = 0$ and conditional second moment at most $P^2/4$. The randomized output is

$$\mathsf{KSwitch}^{\circ}((b,a), \mathrm{KSK}_{z\to z'}) = \big(b + \mathsf{ModDown}^{\circ}(U_B), \mathsf{ModDown}^{\circ}(U_A)\big). \tag{12}$$

All digit and ModDown randomness is sampled anew for each logical source and never cached with the KSK; the same keys may nevertheless be reused across rounds.

Shared correctness interface. Both variants satisfy $d_\lambda \equiv a \pmod{q_\lambda}$. Writing ρ_B, ρ_A for the corresponding Full or Sparse ModDown remainders, their output obeys

$$b' + a'z' = b + az + \frac{\sum_\lambda d_\lambda e^{\mathrm{KS}}_\lambda - \rho_B - \rho_A z'}{P} \pmod{Q}. \tag{13}$$

The quotient is interpreted through the exact R_{QP} congruence and its specified integer lifts. This identity defines the key-switching error; its moment and lifetime bounds are established in Section 5. In the rest of the protocol, $\mathsf{KSwitch}$ means the profile-specific variant just defined, including $\mathsf{KSwitch}^{\circ}$ in the Sparse profile.

4.4 Sparse Profile

Fix one attempt of a Sparse round. We omit t from round-local ciphertexts and describe selection, upload, and encrypted aggregation in that order.

Chunk Selection and Label Admission Sparse employs error feedback by forming $u^{(t)}_i = \delta^{(t)}_i + r^{(t)}_i$, where $r^{(t)}_i$ is the residual from previously omitted updates. With chunk length $B = N/k$, let $J = \lceil d/B \rceil$ and $K_\rho = \max\{1, \lceil \rho d/B \rceil\}$. After zero-padding to length JB, client i selects the K_ρ chunks with the largest ℓ_2 norms, breaking ties by smaller chunk index. Let L_i denote their indices.

To conceal the original chunk indices while allowing cross-client matching, all clients derive a common attempt-specific permutation π_t from K_{perm}, the setup and execution identifiers, and J. Each client sends a bitmap of its permuted labels $Y_i = \{\pi_t(\ell) : \ell \in L_i\}$.

Attempt-specific label generation. The client-only shared key K_{perm} is expanded into the attempt-specific shuffle seed

$$\kappa_t = \mathrm{HMAC\text{-}SHA256\text{-}128}_{K_{\mathrm{perm}}}(\texttt{IDX-PERM} \| \mathsf{setup_id} \| \mathsf{BE}(\mathsf{rid}_t) \| \mathsf{BE}(J)),$$

where the notation $\|$ denotes unambiguous byte concatenation. The parties use the domain-separated SHAKE128 stream tagged `IDX-FY` and a rejection-sampled Fisher–Yates shuffle to derive the same permutation of $\{0, \ldots, J-1\}$. At each shuffle step with range size m, 64-bit words w are rejected unless $w < \lfloor 2^{64}/m \rfloor m$; the accepted index is $w \bmod m$. For $J = 1$, the permutation is the identity. This procedure is an exact finite-string specification, not a separate guarantee of index secrecy against colluding clients.

The server determines the contributor sets $I_{t,y} = \{i \in \mathcal{C} : y \in Y_i\}$ and admits only labels in $Y_t = \{y : |I_{t,y}| \geq \tau\}$. It broadcasts the admitted-label bitmap to the clients.

Exact sparse support and sampling conventions. In the normative Sparse specification, the client-only index key $K_{\text{perm}} \in \{0,1\}^{128}$ determines an attempt-specific permutation via domain-separated HMAC/SHAKE input and a rejection-sampled Fisher–Yates shuffle. A support bitmap occupies $\lceil J/8 \rceil$ bytes in MSB-first order, with unused padding bits set to zero. Index-key distribution assumes an authenticated, confidential client-only setup channel, which is not part of the fixed-cohort simulation theorem's wire view. All private quantizer coins are independently sampled from the exact uniform distribution on $\mathbb{Z}_Q$, or their implementation discrepancy is charged to the global sampler-error budget.

Interleaved Packing and Compressed Upload Client i determines its upload support $H_i^{\text{up}} = Y_i \cap Y_t$ and sends its bitmap to the server. We use the interleaved plaintext layout of M2R (Section 2.4), which places up to k length-B chunks in a single RLWE plaintext. Each *packing lane* is a set of coefficient positions: lane h consists of positions $kx + h$ for $0 \leq x < B$. The admitted labels are sorted and grouped into $R_i = \lceil |H_i^{\text{up}}|/k \rceil$ packs, each with up to k lanes. For pack r, let $\ell_{i,r} \leq k$ be the number of occupied lanes, $y_{i,r,h}$ the label in lane h, and $\ell_{i,r,h} = \pi_t^{-1}(y_{i,r,h})$ its original chunk index.

Client i encodes its update as $\bar{u}_i = \lfloor \Delta u_i^{(t)} \rceil \bmod Q$ and constructs the interleaved plaintext

$$v_{i,r}[kx + h] = \begin{cases} \bar{u}_i[\ell_{i,r,h} B + x], & h < \ell_{i,r}, \\ 0, & h \geq \ell_{i,r}, \end{cases}$$

for $0 \leq x < B$ and $0 \leq h < k$.

For each pack, client i samples a fresh 128-bit seed $\zeta_{i,r}$ and reconstructs $a_{i,r} = \mathsf{UPoly}_Q(\zeta_{i,r}, \texttt{ENC-SP-A} \| \mathsf{setup_id} \| \mathsf{BE}(\mathsf{rid}_t) \| \mathsf{BE64}(i) \| \mathsf{BE64}(r))$. It encrypts under its local key x_i as

$$b_{i,r} = v_{i,r} - a_{i,r} x_i + \epsilon_{i,r}^{\text{enc}} \pmod{Q},$$

where $\epsilon_{i,r}^{\text{enc}} \leftarrow \chi_e$.

To compress the b-component, we use a normalized stochastic quantizer. For $K = 2^c$ and independent $\omega \leftarrow U(\mathbb{Z}_Q)$, define

$$\begin{aligned} \mathsf{SQ}_{K,\omega}(z) &= \left\lfloor \frac{K[z]_Q + Q - 1 - \omega}{Q} \right\rfloor \bmod K, \\ \mathsf{DRec}_K(c) &= \left\lfloor \frac{Qc}{K} + \frac{1}{2} \right\rfloor \bmod Q. \end{aligned} \tag{14}$$

This quantization yields unbiased reconstruction before integer rounding, facilitating the statistical security analysis.

With $K_1 = 2^{c_1}$, client i transmits $\zeta_{i,r}$ and the c_1-bit codes $C^{(1)}_{i,r}[j]$ obtained by applying $\mathsf{SQ}_{K_1,\omega^{(1)}_{i,r,j}}$ to $b_{i,r}[j]$, only for indices $j = kx + h$ with $0 \le x < B$ and $0 \le h < \ell_{i,r}$. Thus, only $B\ell_{i,r}$ of the N b-coefficients are transmitted. The server reconstructs $a_{i,r}$ from the seed, decodes the transmitted coefficients using DRec_{K_1}, and zero-fills the others to obtain $\widehat{b}_{i,r}$.

Key Switching, R2M, and Label-Wise Aggregation The server switches each uploaded ciphertext from x_i to $s_{\mathcal{C}}$ using the cached switching key:

$$ct^{\mathrm{col}}_{i,r} = \mathsf{KSwitch}\big((\widehat{b}_{i,r}, a_{i,r}), \mathrm{KSK}_{i\to\mathcal{C}}\big).$$

The Sparse profile uses the fresh randomized digits and stochastic ModDown of Section 4.3; Full uses deterministic HYBRID switching. The two variants must not be interchanged in the security analysis.

For $k > 1$, the server computes $(ct^{\mathrm{M}}_{i,r,h})_{h=0}^{k-1} = \mathsf{R2M}_k(ct^{\mathrm{col}}_{i,r})$. Since R2M separates the interleaved lanes, zero-filling omitted b-coefficients does not affect the occupied lanes. For $k = 1$, we set $ct^{\mathrm{M}}_{i,r,0} = ct^{\mathrm{col}}_{i,r}$.

For each admitted label $y \in Y_t$, the server computes

$$\mathrm{CT}_y = \sum_{\substack{i\in\mathcal{C},\, r, h:\\ h<\ell_{i,r},\ y_{i,r,h}=y}} \alpha_i \, ct^{\mathrm{M}}_{i,r,h}. \tag{15}$$

These label-wise aggregates are subsequently repacked into RLWE ciphertexts.

M2R Repacking For $k > 1$, the server sorts admitted labels in Y_t and partitions them into groups of at most k labels. Let $y_{g,0} < \cdots < y_{g,\ell_g-1}$ be the labels in group g, with each label occupying one lane; pad all unused lanes by zero MLWE tuples. Denote the resulting rank-k column components by $c_{g,h,j}$, where $0 \le h < k$ and $0 \le j \le k$, and combine each column as in Section 2.4:

$$C_{g,j} = \mathsf{Combine}(c_{g,0,j}, \ldots, c_{g,k-1,j}), \qquad 0 \le j \le k.$$

Before switching, the packed expression is $C_{g,k} + \sum_{j<k} C_{g,j}\Psi_j(s_{\mathcal{C}})$. The server uses the cached keys $\mathrm{swk}^{\mathrm{M2R}}_{j,\mathcal{C}}$ and the Sparse randomized HYBRID routine to

compute

$$\begin{aligned} ct_g^{(0)} &= \mathsf{KSwitch}^{\circ}((C_{g,k}, C_{g,0}), \mathrm{swk}_{0,\mathcal{C}}^{\mathrm{M2R}}), \\ ct_{\mathrm{agg},g} &= ct_g^{(0)} + \sum_{j=1}^{k-1} \mathsf{KSwitch}^{\circ}((0, C_{g,j}), \mathrm{swk}_{j,\mathcal{C}}^{\mathrm{M2R}}). \end{aligned} \tag{16}$$

The output $ct_{\mathrm{agg},g} = (b_g, a_g)$ is an RLWE ciphertext under $t_{\mathcal{C}}$, carrying the interleaved weighted chunks and the M2R switching errors. For $k = 1$, R2M and M2R are both identity paths, and each label aggregate CT_y remains under $s_{\mathcal{C}}$.

4.5 Full and Auxiliary Aggregation

Full-Model Aggregation Unlike Sparse, Full aggregates model updates in a fixed coordinate order without chunk selection or RLWE–MLWE conversion. Each client i divides its model update $\delta_i^{(t)}$ into $R_{\mathrm{model}} = \lceil d/N \rceil$ blocks. For $rN + x < d$, the encoded coefficients are $v_{i,r}[x] = \lfloor \Delta \delta_i^{(t)}[rN + x] \rceil \bmod Q$; the remaining coefficients are zero-padded.

Each block is encrypted under x_i using the seeded encryption and stochastic quantization defined in Section 4.4. The client uploads a fresh seed and N c_1-bit b-component codes per block.

The server reconstructs the ciphertexts, switches them to $s_{\mathcal{C}}$ using $\mathrm{KSK}_{i\to\mathcal{C}}$, and computes $ct_{\mathrm{agg},r} = \sum_{i\in\mathcal{C}} \alpha_i ct_{i,r}^{\mathrm{col}}$.

Auxiliary-State Aggregation Both profiles process auxiliary-state updates through the same dense RLWE aggregation path as Full. Each client computes $\delta_{i,\mathrm{aux}}^{(t)} = z_i^{(t,E)} - z^{(t)}$ and divides it into $R_{\mathrm{aux}} = \lceil d_{\mathrm{aux}}/N \rceil$ blocks. These blocks are encoded and encrypted as in Full, using a separate scale Δ_{aux}, compression parameter c_1^{aux}, and fresh encryption randomness.

Using the same $\mathrm{KSK}_{i\to\mathcal{C}}$, the server computes $ct_{\mathrm{agg},r}^{\mathrm{aux}} = \sum_{i\in\mathcal{C}} \alpha_i ct_{i,r}^{\mathrm{aux,col}}$ under $s_{\mathcal{C}}$. Auxiliary aggregation requires neither sparsification nor RLWE–MLWE conversion. Algorithm 1 summarizes server-side aggregation for a target client set $\mathcal{S}$, where $\mathcal{S} = \mathcal{C}$ in the fixed-cohort execution.

4.6 Compressed Collective Decryption and State Update

After encrypted aggregation, the final cohort collectively decrypts the model and auxiliary ciphertexts without reconstructing the collective secret key. Let $ct_\nu = (b_\nu, a_\nu)$ denote an aggregate ciphertext, where ν identifies a Sparse group or a Full/auxiliary block. Each decryption instance consists of a compressed server request and partial-decryption responses from all finalized participants (equal to $\mathcal{C}$ in the fixed-cohort execution). The following description uses model-path parameters; the auxiliary path uses its own parameter tuple.

Algorithm 1 Server-Side Encrypted Aggregation

Require: Profile p, client set $\mathcal{S}$, uploaded ciphertexts, switching keys, weights α_i, and admitted labels Y (Sparse).
Ensure: Encrypted model and auxiliary aggregates.

for all model uploads (i, r) with $i \in \mathcal{S}$ **do**
  Regenerate $a_{i,r}$ from $\zeta_{i,r}$ and reconstruct $\widehat{b}_{i,r}$ from $C^{(1)}_{i,r}$ (zero-fill omitted coefficients in Sparse).
  $ct^{\mathrm{col}}_{i,r} \leftarrow \mathsf{KSwitch}((\widehat{b}_{i,r}, a_{i,r}), \mathrm{KSK}_{i\to\mathcal{S}})$.
  if $p = \mathsf{Sparse}$ and $k > 1$ **then**
    $(ct^{\mathrm{M}}_{i,r,h})_{h<k} \leftarrow \mathsf{R2M}_k(ct^{\mathrm{col}}_{i,r})$.
  end if
end for
if $p = \mathsf{Sparse}$ **then**
  if $k = 1$ **then**
    $ct^{\mathrm{M}}_{i,r,0} \leftarrow ct^{\mathrm{col}}_{i,r}$ for all model uploads (i, r).
  end if
  for all $y \in Y$ **do**
    Compute CT_y using (15).
  end for
  Sort Y and group labels into batches of at most k.
  for all group g **do**
    if $k > 1$ **then**
      $ct_{\mathrm{agg},g} \leftarrow \mathsf{M2R}_k((\mathrm{CT}_{y_{g,h}})_{h<\ell_g})$.
    else
      $ct_{\mathrm{agg},g} \leftarrow \mathrm{CT}_{y_{g,0}}$.
    end if
  end for
else
  $ct_{\mathrm{agg},r} \leftarrow \sum_{i\in\mathcal{S}} \alpha_i ct^{\mathrm{col}}_{i,r}$ for each model block r.
end if
Reconstruct and key-switch all auxiliary uploads under $s_{\mathcal{S}}$.
$ct^{\mathrm{aux}}_{\mathrm{agg},r} \leftarrow \sum_{i\in\mathcal{S}} \alpha_i ct^{\mathrm{aux,col}}_{i,r}$ for each auxiliary block r.
return Encrypted model and auxiliary aggregates.

Compressed Decryption Request The server compresses each aggregate a-component into c_a-bit codes. Set $K_a = 2^{c_a} \le Q$ and fix the half-up nearest-integer rule of Section 2.2. Both profiles reconstruct a code $y \in \mathbb{Z}_{K_a}$ as

$$\mathsf{DRec}_{K_a}(y) = \lfloor Qy/K_a \rceil \pmod{Q}.$$

Full deterministic aggregate-a compression. For $x \in \mathbb{Z}_Q$, transmit $\mathsf{Comp}^{\mathrm{F}}_{c_a}(x) = \lfloor K_a[x]_Q/Q \rceil \bmod K_a$. This follows the deterministic quantization specified for Full.

Sparse unbiased aggregate-a compression. The Sparse profile, including its auxiliary path, uses an exact randomized grid compressor. Define

$$g_j = \lfloor Qj/K_a \rceil, \quad 0 \le j \le K_a, \qquad g_0 = 0,\ g_{K_a} = Q.$$

For $x = [a]_Q$ find the unique cell j with $g_j \le x < g_{j+1}$. Let $\Delta_j = g_{j+1} - g_j$ and $r = x - g_j$, draw $\omega \leftarrow U(\{0, \dots, \Delta_j - 1\})$, and transmit

$$\mathsf{Comp}^{\circ}_{c_a}(x) = (j + \mathbf{1}_{\{\omega < r\}}) \bmod K_a. \tag{17}$$

To interpret the reconstruction error, the upper code $K_a \equiv 0 \pmod{K_a}$ in the final cell is lifted to the grid point Q rather than 0; the wire code and residue-class decoder are unchanged. This lifted reconstruction has conditional mean x and variance at most $\Delta_j^2/4$. Private coins are fresh for each aggregate coefficient and independent of the subsequent distributed dither.

Using the profile-specific compressor, the server sends $\bar{a}_\nu = \mathsf{Comp}_{c_a}(a_\nu)$; each finalized client reconstructs $\widehat{a}_\nu = \mathsf{DRec}_{K_a}(\bar{a}_\nu)$. The exact integer-lift error bounds are established in Section 5.

Only after fixing the complete collection of aggregate-a codes does the server generate a fresh public 128-bit distributed-dither (DD) seed seed_t^D. From this seed, all parties derive domain-separated public polynomials $\mathsf{Off}_\nu \in R_Q$ and $D_{i,\nu} \in R_Q$ for each decryption instance and client i. For a decryption instance ν and path path_ν, write

$$\mathsf{ctx}_\nu = \mathsf{setup_id} \| \mathsf{BE}(\mathsf{rid}_t) \| \mathsf{path}_\nu \| \mathsf{BE}(\nu).$$

The public polynomials are domain-separated as

$$\mathsf{Off}_\nu = \mathsf{UPoly}_Q(\mathsf{seed}_t^D, \texttt{C2-OFFSET} \| \mathsf{ctx}_\nu),$$
$$D_{i,\nu} = \mathsf{UPoly}_Q(\mathsf{seed}_t^D, \texttt{C2-DITH} \| \mathsf{ctx}_\nu \| \mathsf{BE}(i)).$$

Here UPoly_Q denotes the domain-separated unbiased uniform-polynomial reconstruction from an XOF stream; its exact sampler and failure accounting are treated in the technical foundations. The seed is fresh for each finalized attempt and is generated only after all aggregate-a serialization bytes have been fixed.

Dithered Partial-Decryption Responses Each client i uses $\xi_{i,\nu} = t_i$ for Sparse-model decryption with $k > 1$, and $\xi_{i,\nu} = s_i$ for Full, Sparse with $k = 1$, and auxiliary decryption. It samples fresh smudging noise $g_{i,\nu} \leftarrow D^N_{\mathbb{Z},\sigma_\nu}$ and computes the dithered partial-decryption value

$$X_{i,\nu} = \widehat{a}_\nu \xi_{i,\nu} + g_{i,\nu} + D_{i,\nu} + \mathbf{1}_{\{i=o^\star\}} \mathsf{Off}_\nu \pmod{Q},$$

where $o^\star = \min \mathcal{C}$ adds the public offset once in the fixed-cohort execution. A survivor-specific version uses the minimum finalized survivor. The public dithers randomize the inputs to the response quantizer, while smudging protects the collective secret shares.

Using the stochastic quantizer in (14), client i sends the c_2-bit code $C^{(2)}_{i,\nu}[j] = \mathsf{SQ}_{2^{c_2}, \omega^{(2)}_{i,\nu,j}}(X_{i,\nu}[j])$ for each required coefficient, using fresh private quantization randomness.

For Full and auxiliary blocks, all N coefficients are returned. For a Sparse group g with ℓ_g occupied packing lanes, responses contain only coefficients $j = kx + h$, where $0 \le x < B$ and $0 \le h < \ell_g$. Every client responds for all admitted labels, regardless of its own contribution.

Code-Sum Reconstruction For each returned coefficient, the server first sums the received codes modulo $K_2 = 2^{c_2}$ and then reconstructs the sum only once:

$$\begin{aligned} S_\nu^{(2)}[j] &= \sum_{i\in\mathcal{C}} C_{i,\nu}^{(2)}[j] \bmod K_2, \\ O_\nu[j] &= b_\nu[j] + \mathsf{DRec}_{K_2}(S_\nu^{(2)}[j]) \\ &\quad - \mathsf{Off}_\nu[j] - \sum_{i\in\mathcal{C}} D_{i,\nu}[j] \pmod{Q}. \end{aligned} \tag{18}$$

Importantly, the server reconstructs the sum of codes, rather than summing individually reconstructed responses. The public offset and dithers cancel modulo Q, yielding the raw aggregate output O_ν.

Result Delivery and State Update The server interprets the reconstructed coefficients using their centered representatives, normalizes them by the fixed aggregation weight and encoding scale, and quantizes them for delivery:

$$q_\nu^d[j] = \left\lfloor \frac{\Delta_d\, \mathsf{Center}_Q(O_\nu[j])}{128\Delta} \right\rceil, \qquad \widetilde{u}_\nu[j] = \frac{q_\nu^d[j]}{\Delta_d}.$$

The auxiliary path uses its corresponding encoding and delivery scales.

For Full, the server concatenates the reconstructed model blocks and removes padding. For Sparse, each reconstructed length-B chunk is associated with its admitted label y, which clients map back to the original chunk index using $\pi_t^{-1}(y)$; unreturned chunks are set to zero.

After successfully recovering all required model and auxiliary outputs, clients update $w^{(t+1)} = w^{(t)} + \widetilde{u}_t$ and $z^{(t+1)} = z^{(t)} + \widetilde{u}_t^{\mathrm{aux}}$. In Sparse, each client also updates its error-feedback residual by clearing only the chunks actually included in the aggregation and retaining the remaining chunks of $u_i^{(t)}$. All state changes are committed only upon successful completion of the round.

4.7 Dropout Handling and Finalization

The preceding subsections describe the protocol for a fixed client set $\mathcal{C}$. We extend it to handle client dropouts by reconfiguring the collective key and reaggregating previously uploaded ciphertexts without retransmission within the same execution attempt.

Survivor-set and admission revision. Let $\mathcal{S}_{t,v} \subseteq \mathcal{C}$ denote the surviving clients at revision v of round t. Revisions only remove clients:

$$\mathcal{C} \supseteq \mathcal{S}_{t,0} \supseteq \cdots \supseteq \mathcal{S}_t^\star, \qquad |\mathcal{S}_t^\star| \geq \tau.$$

For Sparse, each revision updates the contributor and admitted-label sets:

$$\begin{aligned} I_{t,y}^{(v)} &= \{i \in \mathcal{S}_{t,v} : y \in Y_i\}, \\ Y_t^{(v)} &= \{y : |I_{t,y}^{(v)}| \geq \tau\}. \end{aligned}$$

Consequently, $Y_t^{(v+1)} \subseteq Y_t^{(v)}$, allowing the server to reuse previously uploaded ciphertexts with their original packing-lane assignments.

Survivor-specific keys and reaggregation. For each revised survivor set $\mathcal{S}$, the server reconstructs the collective public key $p_{0,\mathcal{S}}$ from cached shares and obtains any missing switching keys $\mathrm{KSK}_{i\to\mathcal{S}}$ from surviving clients, reusing previously generated keys. For Sparse with $k > 1$, the M2R keys are reconstructed from cached shares without additional communication. Using these survivor-specific keys, the server recomputes the aggregates from the original uploaded ciphertexts and discards intermediate results for previous survivor sets.

Finalization before collective decryption. Once the required uploads, keys, and aggregates are ready, the server prepares to finalize the survivor set $\mathcal{S}_t^\star$. For Sparse, it also fixes the admitted labels $Y_t^\star$ and their packing-lane assignments, requiring matching FinalizeAck messages from all prospective survivors in response to a common FinalizeNotice. Missing acknowledgments trigger survivor revision and reaggregation before decryption. For Full, the finalized cohort and aggregate collection are bound to the decryption request. The server then performs collective decryption using only the finalized survivors, as described in Section 4.6.

Failure handling and state consistency. Before collective decryption begins, failed attempts may be retried with fresh execution identifiers and uploads, subject to bounded retry and key-target limits. Once the first decryption request is issued, the survivor set is fixed, and any missing partial-decryption response causes the logical round to abort without updating the model, auxiliary state, or residuals. This prevents repeated decryption of the same updates under different survivor sets, which could reveal individual contributions through differences of aggregates. Training may resume in a subsequent round with fresh local updates. Algorithm 2 summarizes the dropout-capable execution.

4.8 Canonical Reference-Wire Encoding and Decoding

The whole-view theorem includes *transmitted bytes*, not just abstract ring elements. Consequently the reference encoder must not assign two different accepted strings to one protocol object. We impose the following canonical reference-codec contract: all field lists and their lengths are determined by the public profile, path, setup and finalized semantic layout; all fixed-width integer fields have a publicly specified byte order and are decoded without accepting nonminimal aliases. This is a contract for the *mathematical reference wire format*, rather than a claim that an application-specific JSON envelope or a deployed binary was independently certified here.

Lemma 1 (Reference-wire canonicality and suffix replay). *For a fixed versioned reference codec and an admissible finalized Core context, the accepted byte representation of each typed message is unique. Conversely the Section 7 conditional simulator, when its output is restricted to the stipulated value ranges, can serialize its sampled ring values, code sums, and support maps to this unique*

Algorithm 2 Dropout-Capable FL Round

Require: Registered clients $\mathcal{C}$, profile p, cached keys, and bounded retry/target policy.
Ensure: Committed updates or no-update.

for each permitted attempt **do**
  Generate fresh rid_t; collect model/auxiliary uploads and Sparse label announcements.
  Initialize $\mathcal{S}_{t,0}$ and $v \leftarrow 0$.
  while $|\mathcal{S}_{t,v}| \geq \tau$ **do**
    Update $Y_t^{(v)}$ (Sparse) and prepare survivor-specific PK, KSK, and M2R keys.
    if all required uploads and keys are ready **then**
      Compute aggregates using Algorithm 1.
      if $p = \mathsf{Sparse}$ **then**
        Send FinalizeNotice; collect matching FinalizeAcks.
      end if
    end if
    if any required upload, key, or Ack is missing **then**
      Set $\mathcal{S}_{t,v+1} \subsetneq \mathcal{S}_{t,v}$; $v \leftarrow v + 1$.
    else
      Finalize $\mathcal{S}_t^\star$ and $Y_t^\star$ (Sparse).
      Compress aggregate-a, generate fresh DD, and request partial decryptions.
      if all model/auxiliary responses are received **then**
        Reconstruct outputs and commit model, auxiliary, and residual states.
        return Commit.
      else
        Preserve states and close round t.
        return $\mathsf{Abort}_{\mathrm{post}}$.
      end if
    end if
  end while
  Discard attempt-specific state before retry.
end for
Preserve states and close round t.
return $\mathsf{Abort}_{\mathrm{pre}}$.

canonical representation. This assertion presupposes the codec's verified fixed-width word-order/length tables and does not certify an independent application-layer transport envelope.

Proof. Typed length delimiters uniquely parse the header. Fixed limb order and word width uniquely parse a polynomial, and the residue range rejection prevents different integer words from decoding to the same limb value. Pairwise-coprime CRT moduli give a unique polynomial residue modulo Q or QP after reconstruction. The range $0 \leq C < 2^c$ determines the unique fixed-width compression code; the simulator must *encode the code* rather than encode a lifted integer representative that happens to have the same residue. For Sparse, bit positions are injective in $y \in \{0, \ldots, J-1\}$ and zero unused bits eliminate aliases. Sorting the admitted set and taking consecutive groups of at most k determines the la-

Table 1. Canonical reference-wire conditions required by the whole-view proof. Coefficient/limb word ordering and widths are part of the declared reference codec.

Object	Canonical decoder constraint
Setup and seed IDs	Fixed version, typed tags and publicly delimited identifier lengths; no ambiguous concatenation
Ring/CRT polynomial	Fixed limb order, coefficient index order and limb word length; reject residues outside each prime and nonzero unused padding
Quantizer code	Exactly c_1, c_a, c_2 or c_d bits for the stipulated path; reject oversized/noncanonical code strings
Sparse bitmap	Exactly $\lceil J/8 \rceil$ bytes; label y in byte $\lfloor y/8 \rfloor$ at bit $7-(y \bmod 8)$; all unused low bits are zero
Sparse slot map	Sorted admitted labels, deterministic groups of at most k lanes, and the stipulated empty-lane zero padding
DD transcript	Fresh root per finalized attempt; fully delimited ctx_ν and disjoint `C2-OFFSET` and `C2-DITH` addresses

bel/slot map; zero padding rules exclude an alternative nonempty representation of an unused slot. DD root, tags, context, client ID and instance ID determine distinct sampler addresses. The inverse-quantizer and DD lifting steps therefore need only serialize uniquely determined, valid reference objects. No uniqueness is claimed for *invalid* byte sequences: those are rejected before any protocol step.

Proof boundary for wire bytes. The reference uniqueness lemma is conditional on the explicit codec version adopted at setup and its canonical fixed-width serialization tables. The parameter-validation procedure in Section 9.2 checks this mathematical reference contract separately from any deployed network serializer. A concrete deployment must also check its own parser and field-length tables against the reference-codec test vectors. In particular, this manuscript does not invent a separate endianness convention for unspecified transport wrappers.

5 Correctness and Technical Foundations

This section proves the algebraic and probabilistic facts needed by the whole-view hybrid argument. It follows the fixed-cohort, semi-honest execution of Sections 3 and 4. The operational dropout procedure is not included in the simulation-security theorem. All statements involving integer-valued errors use explicit lifts of residues, rather than an unqualified interpretation of arithmetic modulo Q. The distinction between deterministic Full and randomized Sparse HYBRID arithmetic is maintained throughout.

5.1 Ring and Key-Switching Identities

We use the coefficientwise quotient convention specified by Section 4.3. In particular, $\mathbb{Z}[X]$ denotes the integer polynomial representatives of the negacyclic ring,

and a congruence modulo QP can be lifted to an integer identity by adding a multiple of QP. Such an adjustment changes a quotient by P only by a multiple of Q, and therefore does not change its residue in R_Q.

Lemma 2 (CRT reconstruction). *Let $Q = \prod_{\lambda<m} q_\lambda$ and let γ_λ be the CRT idempotents from Section 2.2. For any $a \in R_Q$ and polynomials d_λ satisfying $d_\lambda \equiv a \pmod{q_\lambda}$ coefficientwise,*

$$\sum_{\lambda<m} \gamma_\lambda d_\lambda \equiv a \pmod{Q}.$$

Consequently, the source-secret contribution of a HYBRID switching row is unchanged modulo Q when canonical CRT digits are replaced by signed, randomized digits with the same residues.

Proof. Reducing the sum modulo an arbitrary q_μ leaves only the $\lambda = \mu$ summand, which is congruent to a. The Chinese remainder theorem proves the first assertion. Multiplying by the integer P and by the source secret preserves that congruence, proving the second assertion.

Lemma 3 (Key-switching rows). *Let $f_{\mathcal{C}} = \sum_{i\in\mathcal{C}} f_i$. For the cached key $\mathrm{KSK}_{i\to\mathcal{C}}$,*

$$\begin{aligned} B_{i,\lambda} + A_{i,\lambda} s_{\mathcal{C}} &= P\gamma_\lambda x_i + e^{\mathrm{KS}}_{i,\lambda}, \\ e^{\mathrm{KS}}_{i,\lambda} &= e^0_{i,\lambda} + e^1_{i,\lambda} s_{\mathcal{C}} + u_{i,\lambda} f_{\mathcal{C}} \end{aligned} \pmod{QP}. \tag{19}$$

For Sparse with $k > 1$, the cohort-summed M2R row satisfies

$$\begin{aligned} B^M_{j,\lambda,\mathcal{C}} + a^M_{j,\lambda} t_{\mathcal{C}} &= P\gamma_\lambda \Psi_j(s_{\mathcal{C}}) + e^M_{j,\lambda,\mathcal{C}}, \\ e^M_{j,\lambda,\mathcal{C}} &= \sum_{i\in\mathcal{C}} e^M_{j,\lambda,i} \end{aligned} \pmod{QP}. \tag{20}$$

Neither identity requires reconstructing a collective secret.

Proof. Substitute $p_{0,\mathcal{C}} + p_1 s_{\mathcal{C}} = f_{\mathcal{C}}$ into the definitions of $A_{i,\lambda}$ and $B_{i,\lambda}$. For the M2R key, add the individual $b^M_{j,\lambda,i}$ shares and use the additivity of Ψ_j and $t_{\mathcal{C}} = \sum_i t_i$. These are identities in R_{QP}, not independence assertions about cached material.

Lemma 4 (Cached-row second moments). *Assume independent centered key-generation errors with per-coefficient second moment v_e, independent ternary key coefficients with second moment $2/3$, and the fixed-cohort setup of Section 4. For every local switching-key row and coefficient u,*

$$\mathbb{E}[(e^{\mathrm{KS}}_{i,\lambda}[u])^2] \le v_e\Big(1 + \frac{4Nn}{3}\Big). \tag{21}$$

For the M2R key, the error of one cohort-summed row obeys $\mathbb{E}[(e^M_{j,\lambda,\mathcal{C}}[u])^2] = nv_e$. Both statements concern the setup distribution *of cached rows; rows are not freshly sampled on every use.*

Proof. From Lemma 3, the local-row error is $e^0 + e^1 s_{\mathcal{C}} + u f_{\mathcal{C}}$. Each term is centered, and cross terms vanish under the independent setup samplers. The coefficientwise second moment of $e^1 s_{\mathcal{C}}$ is $v_e \, \mathbb{E}\|s_{\mathcal{C}}\|_2^2 = 2Nnv_e/3$ by the negacyclic convolution identity. Similarly, the second moment of $u f_{\mathcal{C}}$ is $(2/3)\mathbb{E}\|f_{\mathcal{C}}\|_2^2 = 2Nnv_e/3$. The direct e^0 term contributes v_e. The M2R error is a sum of n independent centered error polynomials and hence has per-coefficient second moment nv_e. This calculation does not require the *different* switching-key rows to be mutually independent after they have been cached.

Lemma 5 (Exact sourcewise HYBRID switching). *Consider (b, a) with $b + az = \mu + e_{\text{in}} \pmod{Q}$ and a switching key from z to z' satisfying $B_\lambda + A_\lambda z' = P\gamma_\lambda z + e_\lambda$ modulo QP. Let $d_\lambda \equiv a \pmod{q_\lambda}$, $U_B = \sum_\lambda d_\lambda B_\lambda$, and $U_A = \sum_\lambda d_\lambda A_\lambda$. For either profile-specific ModDown, let ρ_B, ρ_A be its integer remainders, satisfying $\rho_U \equiv U \pmod{P}$. The switched pair is*

$$(b', a') = \big(b + (U_B - \rho_B)/P, (U_A - \rho_A)/P\big) \pmod{Q}$$

and satisfies

$$b' + a'z' = \mu + e_{\text{in}} + e_{\text{sw}} \pmod{Q}, \qquad e_{\text{sw}} = \frac{\sum_\lambda d_\lambda e_\lambda - \rho_B - \rho_A z'}{P} \pmod{Q}. \tag{22}$$

Here the quotient in e_{sw} has an integer polynomial lift and is not an informal division in R_Q.

Proof. Both numerators $U_B - \rho_B$ and $U_A - \rho_A$ are coefficientwise divisible by P. The key-row relation gives

$$U_B + U_A z' = P \sum_\lambda d_\lambda \gamma_\lambda z + \sum_\lambda d_\lambda e_\lambda \pmod{QP}.$$

Choose integer representatives of this congruence. Subtract $\rho_B + \rho_A z'$ and divide by P. The possible added multiple of QP vanishes modulo Q after division. Lemma 2 replaces $\sum_\lambda d_\lambda \gamma_\lambda z$ by az modulo Q. Adding b proves the claimed identity.

5.2 Exact RLWE–MLWE Conversion and Partial Uploads

Lemma 6 (Interleaved negacyclic multiplication). *Let $B = N/k$, $Y = X^k$, and let $a, s \in R_Q$. For each $0 \le h < k$,*

$$\mathsf{Split}_h(as) = \sum_{j=0}^{k-1} Y^{\mathbf{1}\{h<j\}} \mathsf{Split}_{[h-j]_k}(a) \mathsf{Split}_j(s) \quad \text{in } R_{Q,B}. \tag{23}$$

Consequently, the R2M construction in (2) *satisfies* (3) *exactly, without a switching key or additional error.*

Proof. Write $a = \sum_{r<k} X^r a_r(Y)$ and $s = \sum_{j<k} X^j s_j(Y)$ with $a_r = \mathsf{Split}_r(a)$ and $s_j = \mathsf{Split}_j(s)$. A product term has $X^{r+j} a_r(Y) s_j(Y)$. For output lane h, the unique $r \in \{0, \ldots, k-1\}$ with $r + j \equiv h \pmod{k}$ is $r = [h - j]_k$. When $j > h$, one factor $X^k = Y$ remains; otherwise no such factor remains. This proves (23). The relation $Y^B = -1$ already accounts for negacyclic wrapping inside each component. Applying the identity to $b + as = \mu + e$ proves the R2M claim.

Lemma 7 (M2R correctness). *Let ct_h^{M} denote up to k MLWE ciphertexts under the same split secret, with unused lanes padded by zero tuples. For column combinations $C_j = \mathsf{Combine}((c_{h,j})_{h<k})$,*

$$C_k + \sum_{j<k} C_j \Psi_j(s) = \mathsf{Combine}((\mu_h^{\mathrm{M}} + e_h^{\mathrm{M}})_{h<k}) \pmod{Q}.$$

Applying the k sourcewise M2R key switches specified by (16) *produces an RLWE pair (b_g, a_g) under $t_{\mathcal{C}}$ satisfying*

$$b_g + a_g t_{\mathcal{C}} = \mathsf{Combine}((\mu_h^{\mathrm{M}} + e_h^{\mathrm{M}})_{h<k}) + \sum_{j=0}^{k-1} e_{g,j}^{M2R} \pmod{Q}, \tag{24}$$

where $e_{g,j}^{M2R}$ is the exact error (22) *for source j with $z = \Psi_j(s_{\mathcal{C}})$ and $z' = t_{\mathcal{C}}$.*

Proof. For each j, multiplication by $\Psi_j(s)$ acts on each interleaved lane as multiplication by $\mathsf{Split}_j(s)$. The first relation follows by combining the k MLWE decryption relations, and is the column identity (4). Apply Lemma 5 to the pair (C_k, C_0) for source $j = 0$ and to $(0, C_j)$ for $j > 0$, all under the same target secret. Adding the k resulting ciphertexts gives (24). The claim uses exactly one ModDown for each source and each output component, performed after the source's full gadget sum; interchanging these operations would define a different protocol.

Lemma 8 (Occupied-lane preservation under partial b upload). *For a local Sparse pack with occupied lanes $H \subseteq \{0, \ldots, k-1\}$, let $P_H : R_Q \to R_Q$ retain exactly the coefficient positions $kx + h$ with $h \in H$, $0 \le x < B$, zeroing the other positions. Then for every $z \in R_Q$ and $h \in H$,*

$$\mathsf{Split}_h(P_H z) = \mathsf{Split}_h(z).$$

After the server zero-fills untransmitted b-coefficients, local key switching followed by R2M has exactly the same occupied-lane b component *as an upload of the entire reconstructed b polynomial. This does not assert equality of unoccupied lanes.*

Proof. The split operation reads only coefficients retained by P_H. For fixed a, KSK and the sampled switching arithmetic, the switched b component is the input $\widehat{b}$ plus a correction determined entirely by the a side and the cached keys; therefore the correction is unchanged by zero-filling other b positions. R2M selects Split_h of this switched component, proving the statement on occupied lanes.

5.3 Exact Quantizers and Integer Error Representatives

All nearest-integer operations here use the protocol's half-up rule $\lfloor x \rceil = \lfloor x+1/2 \rfloor$. The residue $[x]_Q$ and a chosen integer lift of a quantization error are different objects; subtracting two canonical residues can yield a misleading difference near zero. We use exact, fresh uniform coins unless an explicit implementation total-variation discrepancy is charged.

Lemma 9 (Unbiased signed CRT digits). *Fix $q \geq 2$ and $r \in \{0, \ldots, q-1\}$. With $\omega \leftarrow U(\mathbb{Z}_q)$ set $d^\circ = r - q\mathbf{1}_{\{\omega<r\}}$. Then*

$$\begin{aligned} d^\circ &\equiv r \pmod q, \qquad \mathbb{E}[d^\circ \mid r] = 0, \\ \mathbb{E}[(d^\circ)^2 \mid r] &= r(q-r) \leq q^2/4, \qquad |d^\circ| \leq q-1. \end{aligned} \tag{25}$$

The statement applies coefficientwise to every Sparse HYBRID digit, conditional on its already-fixed input residue.

Proof. The indicator equals one with probability r/q. Thus $d^\circ = r$ with probability $1 - r/q$ and $d^\circ = r - q$ with probability r/q. Direct expansion gives the mean, variance, and absolute bound. Congruence is immediate.

Lemma 10 (Unbiased stochastic ModDown). *For $r = [U[u]]_P$ with $0 \leq r < P$, draw fresh ω uniformly from $\mathbb{Z}_P$ and set*

$$\rho^\circ = r - P\mathbf{1}_{\{\omega<r\}}.$$

Then $U[u] - \rho^\circ$ is divisible by P and

$$\mathbb{E}[\rho^\circ \mid U] = 0, \qquad \mathbb{E}[(\rho^\circ)^2 \mid U] = r(P-r) \leq P^2/4.$$

The A- and B-side remainders are generated with independent fresh coins after their respective sourcewise QP gadget sums have been fixed.

Proof. The indicator is Bernoulli with success probability r/P. The moment identities follow as in Lemma 9. Since $\rho^\circ \equiv U[u] \pmod P$, integer division is exact. Independent coin generation is part of the protocol, not a consequence of the moment calculation.

Lemma 11 (Full and Sparse aggregate-a grid errors). *For $K_a = 2^{c_a} \leq Q$, define the increasing grid $g_j = \lfloor Qj/K_a \rceil$ for $0 \leq j \leq K_a$ and $g_0 = 0$, $g_{K_a} = Q$. Put $G_a = \max_{0 \leq j < K_a}(g_{j+1} - g_j)$. For Sparse, condition on the coefficient $x = [a]_Q \in [g_j, g_{j+1})$. The lifted randomized decoder $\widetilde{a} \in \{g_j, g_{j+1}\}$ satisfies*

$$\begin{aligned} d_a^{\mathbb{Z}} &= \widetilde{a} - x, & \mathbb{E}[d_a^{\mathbb{Z}} \mid x] &= 0, \\ \mathbb{E}[(d_a^{\mathbb{Z}})^2 \mid x] &= (x - g_j)(g_{j+1} - x) \leq G_a^2/4, & G_a &\leq \lceil Q/K_a \rceil. \end{aligned} \tag{26}$$

For Full, the nearest-code deterministic compressor has an integer-lift error $d_{a,F}^{\mathbb{Z}}$ satisfying

$$|d_{a,F}^{\mathbb{Z}}| \leq E_a(c_a) := \left\lceil Q/2^{c_a+1} + 1/2 \right\rceil,$$

but it is not *claimed to be conditionally centered. For the final Sparse cell, $g_{K_a} = Q$ is the integer lift of transmitted code zero and has the same residue modulo Q.*

Proof. Write $r = x - g_j$ and $\Delta_j = g_{j+1} - g_j$. The upper endpoint is selected with probability r/Δ_j, giving conditional expectation x and variance $r(\Delta_j - r)$. The grid spacing is at most $\lceil Q/K_a \rceil$. For Full, nearest-code rounding places x within half a quantization interval, with at most one additional half-integer rounding deviation for the grid endpoints; taking the displayed ceiling gives the stated conservative bound. All equalities are first applied to integer lifts, then reduced modulo Q.

Lemma 12 (Exact normalized stochastic c_1/c_2 code law). *Let $K = 2^c$ and write $Kx = Qb + r$ with $x = [z]_Q$ and $0 \le r < Q$. For $\omega \leftarrow U(\mathbb{Z}_Q)$, the wire code $C = \mathsf{SQ}_{K,\omega}(z)$ has distribution*

$$C = (b + \beta) \bmod K, \qquad \Pr[\beta = 1 \mid x] = r/Q.$$

Define the small integer representative

$$\eta_K^{\mathbb{Z}} = \left\lfloor \frac{Q\beta - r}{K} \right\rceil. \tag{27}$$

Then the independent reconstruction of this code satisfies

$$\mathsf{DRec}_K(C) - x \equiv \eta_K^{\mathbb{Z}} \pmod{Q}, \quad \left| \eta_K^{\mathbb{Z}} - \frac{Q\beta - r}{K} \right| \le \tfrac{1}{2}. \tag{28}$$

The unrounded *term $(Q\beta - r)/K$ has conditional mean zero and lies in an interval of length Q/K. The rounded representative need not have exactly zero conditional mean; its rounding remainder is at most $1/2$. For c_1, reconstruction is performed* per transmitted coefficient*; for c_2, codes are summed before the one reconstruction specified in* (18)*.*

Proof. The defining floor gives $C = b$ unless $\omega < r$, in which case it gives $b + 1$. Before reducing C modulo K, the reconstructed integer is $\lfloor Q(b + \beta)/K \rceil$. Since $Q(b + \beta)/K = x + (Q\beta - r)/K$ and x is an integer, translation invariance of half-up rounding implies

$$\lfloor Q(b + \beta)/K \rceil - x = \lfloor (Q\beta - r)/K \rceil .$$

Reducing C modulo K changes its reconstruction by a multiple of Q, proving the residue identity. Conditional centering of the unrounded term follows from $\mathbb{E}[\beta \mid r] = r/Q$; the final bound is the nearest-integer rounding guarantee.

5.4 Exact Decryption, Dither Cancellation, and Reconstruction

For an aggregate instance ν, the model/auxiliary secret in the selected decryption domain is

$$z_{\mathcal{C},\nu} = \sum_{i \in \mathcal{C}} \xi_{i,\nu} = \begin{cases} t_{\mathcal{C}}, & \text{Sparse model, } k > 1, \\ s_{\mathcal{C}}, & \text{Full, Sparse } k = 1, \text{ or auxiliary.} \end{cases}$$

Write $\bar{a}_\nu$ for the server's aggregate-a code, $\widehat{a}_\nu$ for its decoded residue, and $d_\nu = \widehat{a}_\nu - a_\nu \in R_Q$. Define the pre-$c_2$ raw response $h_{i,\nu} = \widehat{a}_\nu \xi_{i,\nu} + g_{i,\nu}$.

Lemma 13 (Pre-c_2 phase identity). *Suppose an aggregate ciphertext satisfies $b_\nu + a_\nu z_{\mathcal{C},\nu} = M_\nu + U_{1,\nu}^{\mathrm{all}} + E_\nu^{\mathrm{nat},0} \pmod Q$, where M_ν is the encoded, error-free weighted aggregate, $U_{1,\nu}^{\mathrm{all}}$ is the sum of lifted c_1 reconstruction errors transported to the released lanes, and $E_\nu^{\mathrm{nat},0}$ collects encryption and switching errors other than U_1. Then*

$$\begin{aligned} T_\nu &:= b_\nu + \sum_{i\in\mathcal{C}} h_{i,\nu} \\ &= M_\nu + U_{1,\nu}^{\mathrm{all}} + \sum_{i\in\mathcal{C}} g_{i,\nu} + \Delta_\nu \pmod Q, \\ \Delta_\nu &:= E_\nu^{\mathrm{nat},0} + d_\nu z_{\mathcal{C},\nu}. \end{aligned} \tag{29}$$

The term $d_\nu z_{\mathcal{C},\nu}$ contains the whole *collective secret, including corrupted shares; it must not be silently absorbed into corrupted-client data.*

Proof. Substitute $\sum_i h_{i,\nu} = \widehat{a}_\nu z_{\mathcal{C},\nu} + \sum_i g_{i,\nu}$ and $\widehat{a}_\nu = a_\nu + d_\nu$ into the ciphertext phase. No independence assumption is used.

Lemma 14 (Byte-compatible distributed-dither cancellation). *For each released coefficient, let $X_{i,\nu} = h_{i,\nu} + D_{i,\nu} + \mathbf{1}_{\{i=o^\star\}}\mathsf{Off}_\nu \pmod Q$ and $C_{i,\nu}^{(2)} = \mathsf{SQ}_{K_2,\omega_i}(X_{i,\nu})$. Define the* exact realized *code-sum error in R_Q by*

$$Z_{2,\nu} = \mathsf{DRec}_{K_2}\left(\sum_i C_{i,\nu}^{(2)} \bmod K_2\right) - \sum_i X_{i,\nu} \pmod Q.$$

Then reconstruction yields

$$O_\nu = b_\nu + \sum_i h_{i,\nu} + Z_{2,\nu} \pmod Q. \tag{30}$$

Thus public dither does not itself contribute additive noise to the decoded aggregate. The c_2 error is the one code-sum reconstruction error*, not a sum of separately decoded client errors.*

Proof. Insert $\sum_i X_{i,\nu} = \sum_i h_{i,\nu} + \sum_i D_{i,\nu} + \mathsf{Off}_\nu$ into the reconstruction rule (18); the offset and all per-client dithers cancel exactly modulo Q.

5.5 Integer Lifts and Numerical Correctness

Lemma 15 (Integer lift of the flooding shift). *For every completed Full or Sparse decryption instance, the protocol admits integer-polynomial representatives $E_\nu^{\mathrm{nat},0,\mathbb{Z}}$ and $d_\nu^{\mathbb{Z}}$ consistent with the realized coefficient arithmetic such that*

$$\Delta_\nu^{\mathbb{Z}} = E_\nu^{\mathrm{nat},0,\mathbb{Z}} + d_\nu^{\mathbb{Z}} z_{\mathcal{C},\nu} \in \mathbb{Z}[X]/(X^N+1), \qquad \Delta_\nu^{\mathbb{Z}} \bmod Q = \Delta_\nu. \tag{31}$$

For Sparse $k > 1$, $E_\nu^{\mathrm{nat},0,\mathbb{Z}}$ includes the per-source M2R errors, whereas Full and Sparse $k = 1$ have no M2R term. The lift of aggregate-a compression treats the last grid endpoint Q as an integer before reduction modulo Q.

Proof. For each source key switch, choose integer representatives of the key rows and their gadget sums. Because the ModDown remainders have the same residues modulo P as the corresponding sums, both quotients are exact integer polynomials. Lemma 5 lifts the switching error; the additional multiples of QP in its source identity produce multiples of Q after division. Encryption errors, the realized c_1 lifts from Lemma 12, integer aggregation weights, R2M, and Combine preserve integer polynomial representatives. Sparse M2R is a finite sum of the same sourcewise key switches. For $d_\nu^{\mathbb{Z}}$ take the corresponding Full nearest-grid or Sparse randomized-grid integer error from Lemma 11. All representatives reduce to the realized residues in (29), proving the statement.

Proposition 1 (Conditional numerical reconstruction). *Fix a released model coordinate and let $u^\star$ be the ideal real-valued weighted update formed with the* same *approved contributor set and the fixed weights $\alpha_i/128$. Assume the integer input encoding for each contributor has error at most $1/2$ at scale Δ, the total remaining realized integer error (including c_1, natural switching/compression, smudging, and c_2) has magnitude at most B_{num}, and no modular wrap or final-delivery clipping occurs. Then the delivered update $\widetilde{u}$ obeys*

$$|\widetilde{u} - u^\star| \le \frac{\sum_{i\in I} \alpha_i/2 + B_{\mathrm{num}}}{128\Delta} + \frac{1}{2\Delta_d}, \tag{32}$$

where I is the released coordinate's contributor set ($I = \mathcal{C}$ in Full). For the auxiliary path, replace the scales and error bound by their auxiliary counterparts. This is an implication conditioned on numerical bounds; it is not by itself a probability certificate for those bounds.

Proof. The difference between the exact scaled encoding and the ideal scaled update has absolute value at most $\sum_{i\in I} \alpha_i/2$. By the no-wrap hypothesis, centering the decoded residue recovers the corresponding signed integer. Dividing by 128Δ yields the first term; final nearest-integer delivery contributes at most $1/(2\Delta_d)$. Since contributors are not renormalized, the denominator remains 128 even when $I \neq \mathcal{C}$.

5.6 Conditional Moments and Adaptive Sparse Selection

The Sparse protocol allows training updates and selected supports to depend on previously released aggregates. Moreover, one KSK or M2R key may be reused throughout the execution, so later selection events can be correlated with cached errors. Our estimates will not assume that selected supports are independent of cached keys. Instead, we condition on the information available just *before* each fresh arithmetic random choice, and use pathwise bounds on the number of selected/released coordinates.

Lemma 16 (Fresh centered convolution). *Let $\mathcal{F}$ be a sigma-field and let $z \in R_Q$ be fixed conditional on $\mathcal{F}$. Suppose the fresh integer coefficients of d are*

conditionally independent, with $\mathbb{E}[d_j \mid \mathcal{F}] = 0$ *and* $\mathbb{E}[d_j^2 \mid \mathcal{F}] \le v_d$. *For every coefficient* u *of the negacyclic product,*

$$\mathbb{E}[(dz)[u]^2 \mid \mathcal{F}] \le v_d \, \|z\|_2^2. \tag{33}$$

The same assertion applies when z *is a function of cached secrets, cached key-row errors, and the entire previously released transcript, provided* z *is fixed before* d *is sampled.*

Proof. Negacyclic convolution writes $(dz)[u]$ as a signed sum of $d_j z_{u-j}$. Conditional centering and independence annihilate all off-diagonal terms in its squared expansion. Bounding each remaining variance by v_d gives the result.

Lemma 17 (Martingale orthogonality). *Consider a chronological sequence of integrable, square-integrable scalar increments* Z_r *adapted to a filtration* $(\mathcal{F}_r)$ *with* $\mathbb{E}[Z_r \mid \mathcal{F}_{r-1}] = 0$. *Then*

$$\mathbb{E}\Big[\big(\sum_r Z_r\big)^2\Big] = \sum_r \mathbb{E}[Z_r^2]. \tag{34}$$

The result applies coefficientwise (and then after summing coordinates) to Sparse fresh signed-digit, stochastic ModDown, and unbiased aggregate-a increments, provided their chronological centering is established with the relevant cached data and adaptive history included *in the conditioning field. It does not identify the deterministic Full errors as martingale increments.*

Proof. For $r < s$, Z_r is $\mathcal{F}_{s-1}$-measurable, so $\mathbb{E}[Z_r Z_s] = \mathbb{E}[Z_r \mathbb{E}[Z_s \mid \mathcal{F}_{s-1}]] = 0$. Expand the square and sum.

Lemma 18 (Pathwise incidence bound without independence). *Let* $I_r \in \{0,1\}$ *be any random indicators, potentially dependent on cached state, such that* $\sum_r I_r \le L$ *holds on every execution path. For every nonnegative cached random variable* X,

$$\mathbb{E}\Big[\sum_r I_r X\Big] \le L \, \mathbb{E}[X]. \tag{35}$$

Proof. The deterministic inequality $\sum_r I_r X \le LX$ holds on every sample path; take expectations.

Ordering of fresh randomness. For Sparse source key switches, the gadget digits and ModDown operands are functions of the source a side and the cached rows; the occupied uploaded b-coefficients enter as additive terms only. The M2R gadget inputs are also formed from R2M a-columns. Hence the fresh server-side digit/ModDown coins may be drawn before revealing the honest clients' c_1 suffix coins without changing the joint distribution, as long as the same digit is used for both key components and all sourcewise ordering rules are preserved. This is a reordering of independent random tapes, *not* a claim that the cached keys or adaptive supports are mutually independent.

5.7 Whole-Execution Error Energy for the Two Profiles

The quantitative bounds below use the N-coefficient ring of Section 2.2, independent ternary keys of variance 2/3, exact centered encryption/key-generation Gaussian samples with per-coefficient second moment v_e, and the specified independent coins for each Sparse arithmetic use. They describe expected integer-lift squared errors over fixed-cohort executions; they do not condition on a high-probability setup-good event.

Sparse direct, M2R, and aggregate-a energies. Let

$$\begin{aligned} G_{\mathrm{ZM}}^2 &:= \frac{N}{4P^2}\sum_{\lambda<m} q_\lambda^2, \\ V_{\mathrm{MD}}(n) &:= \frac{1}{4}+\frac{Nn}{6}, \\ V_{\mathrm{loc}}^{\mathrm{ZM}}(n) &:= v_e + G_{\mathrm{ZM}}^2 v_e\Big(1+\frac{4Nn}{3}\Big) + V_{\mathrm{MD}}(n), \\ V_{\mathrm{M2R}}^{\mathrm{ZM}}(n) &:= k\big(G_{\mathrm{ZM}}^2 n v_e + V_{\mathrm{MD}}(n)\big), \\ V_{ca}^{\mathrm{ZM}}(p) &:= \frac{g_a(c_a^p)^2 nN}{6}, \\ g_a(c) &:= \max_j (g_{j+1}-g_j). \end{aligned} \tag{36}$$

These constants are obtained by expanding the cached KSK row errors using Lemma 3, applying the fresh-digit and ModDown moments of Lemmas 9–10, and using Lemma 16 on each negacyclic product. In particular, the cached local-row coefficient energy bound follows from Lemma 4. The aggregate-a term uses $\mathbb{E}\|s_{\mathcal{C}}\|_2^2 = 2Nn/3$, or the same identity with $t_{\mathcal{C}}$, and $\mathbb{E}[d_a^2 \mid a] \le g_a^2/4$.

For Sparse let $W_2^2 = \sum_{i\in\mathcal{C}} \alpha_i^2$, $R_a = \lceil d_{\mathrm{aux}}/N \rceil$, and

$$\begin{aligned} J &= \lceil d/B \rceil, \qquad Y_{\max} = \min\{J, \lfloor nK_\rho/\tau \rfloor\}, \\ L_m &= BTY_{\max}, \qquad L_a = NTR_a. \end{aligned} \tag{37}$$

A client contributes at most $TK_\rho B$ model coordinates across T rounds, while the union of released Sparse model coordinates has cardinality at most L_m (counting repeated rounds). These are *pathwise* bounds, irrespective of adaptive support.

Lemma 19 (Setup-measurable dominating envelopes for Sparse). *Let* $e_{i,\lambda}^{\mathrm{KS}}$ *and* $e_{j,\lambda,\mathcal{C}}^{M}$ *denote the* underlying pre-modulus integer error polynomials *generated with the setup- cached KSK and cohort M2R rows in Lemma 3, not arbitrary centered lifts of their* R_{QP} *residue classes. Before the current fresh arith-*

metic draws, define the nonnegative setup-measurable variables

$$\begin{aligned} X_i^{\mathrm{dir}} &= v_e + \sum_{\lambda<m} \frac{q_\lambda^2}{4P^2}\|e_{i,\lambda}^{\mathrm{KS}}\|_2^2 + \frac{1+\|s_{\mathcal{C}}\|_2^2}{4}, \\ X^{\mathrm{M2R}} &= \sum_{j<k}\sum_{\lambda<m} \frac{q_\lambda^2}{4P^2}\|e_{j,\lambda,\mathcal{C}}^{M}\|_2^2 + \frac{k(1+\|t_{\mathcal{C}}\|_2^2)}{4}, \\ X_p^{ca} &= \frac{g_a(c_a^p)^2}{4}\|z_{\mathcal{C},p}\|_2^2, \end{aligned} \tag{38}$$

where $z_{\mathcal{C},p} = t_{\mathcal{C}}$ only for Sparse model outputs with $k > 1$, and is $s_{\mathcal{C}}$ otherwise. For every current released coefficient, conditional on the setup and on the history preceding its fresh arithmetic coins, the variance contribution of the corresponding direct source, M2R output, and aggregate-a rounding is bounded by its respective X above. The same *X_i^{dir} dominates every coefficient and every round using source i; X^{M2R} and X_p^{ca} likewise dominate every corresponding released output coordinate. Moreover,*

$$\begin{aligned} \mathbb{E}X_i^{\mathrm{dir}} &\le V_{\mathrm{loc}}^{\mathrm{ZM}}(n), \\ \mathbb{E}X^{\mathrm{M2R}} &\le V_{\mathrm{M2R}}^{\mathrm{ZM}}(n), \\ \mathbb{E}X_p^{ca} &= V_{ca}^{\mathrm{ZM}}(p). \end{aligned} \tag{39}$$

Proof. Condition on *all* setup keys and cached switching rows. For a single source and coefficient, the fresh randomized CRT digits are conditionally independent across their coefficient positions, have mean zero, and have variance at most $q_\lambda^2/4$. The negacyclic product with the fixed row $e_{i,\lambda}^{\mathrm{KS}}$ therefore has conditional variance at most $q_\lambda^2\|e_{i,\lambda}^{\mathrm{KS}}\|_2^2/4$. After the prescribed division by P, this is precisely the corresponding summand in X_i^{dir}. The fresh B- and A-ModDown remainders are centered conditional on the previously constructed numerators, and each has variance at most $P^2/4$. The A-remainder is multiplied by the fixed target secret $s_{\mathcal{C}}$; Lemma 16 then contributes at most $(1+\|s_{\mathcal{C}}\|_2^2)/4$. The fresh local encryption error contributes v_e.

For M2R, a released coefficient receives at most one (possibly negated) coefficient from each of the k module switching sources. Applying the same row-norm bound to all $j < k$ and the M2R ModDown remainders gives X^{M2R}. A fresh unbiased aggregate-a rounding error has variance at most $g_a^2/4$ per input coefficient; convolution with the fixed threshold secret gives X_p^{ca}. These are conditional *variance* bounds for the fresh martingale increments; they do not assume independence between the setup cache and the adaptively selected coefficient.

For the expectations, use the cached-row identities and Lemma 4: $\mathbb{E}\|e_{i,\lambda}^{\mathrm{KS}}\|_2^2 \le Nv_e(1+4Nn/3)$ and $\mathbb{E}\|e_{j,\lambda,\mathcal{C}}^{M}\|_2^2 = Nnv_e$. The independent ternary setup secrets satisfy $\mathbb{E}\|s_{\mathcal{C}}\|_2^2 = \mathbb{E}\|t_{\mathcal{C}}\|_2^2 = 2Nn/3$. Substitution gives exactly the three quantities in (39); the first two are upper bounds. No fresh resampling of cached rows is used.

How the cached envelope is used. The distinction between a *fixed cached quantity* and a *fresh random increment* can be seen directly in (38). For source i, the random variable X_i^{dir} is determined at setup; the indicator $J_{i,r}$ is decided from the current admitted support before the new arithmetic coins for coordinate r are drawn. If $Z_{i,r}$ is the resulting centered direct-path error increment, the two inequalities used later are

$$\begin{aligned} \mathbb{E}[Z_{i,r}^2 \mid \mathcal{F}_{r^-}] \le X_i^{\mathrm{dir}}, \qquad \mathbb{E}[Z_{i,r} \mid \mathcal{F}_{r^-}] = 0, \\ \sum_r J_{i,r} \le T K_\rho B. \end{aligned} \tag{40}$$

Here $\mathcal{F}_{r^-}$ contains the actual setup cache, the admitted support and every arithmetic coin revealed before this increment. The first inequality is conditional on that field; the second holds on every complete execution path. Consequently

$$\begin{aligned} \mathbb{E}\left[\sum_r J_{i,r} Z_{i,r}^2\right] &\le \mathbb{E}\left[X_i^{\mathrm{dir}} \sum_r J_{i,r}\right] \\ &\le T K_\rho B\, \mathbb{E} X_i^{\mathrm{dir}}. \end{aligned} \tag{41}$$

There is no factorization $\mathbb{E}[J_{i,r} X_i^{\mathrm{dir}}] = \mathbb{E}[J_{i,r}]\mathbb{E}[X_i^{\mathrm{dir}}]$: that equality could fail because earlier published model values may correlate later supports with the setup cache. The same argument uses X^{M2R} or X_p^{ca} for the other terms, with at most L_m or L_a released coordinate uses from (37).

Lemma 20 (Fresh-coin filtration with adaptive incidence). *For any released Sparse coefficient r, augment the conditioning field with the complete setup cache, application history, current admitted support, source layout, and current A-side masks. Within the corresponding execution, reveal the fresh local-encryption errors, randomized CRT digits, sourcewise ModDown coins, optional M2R digits/ModDown coins, and finally aggregate-a rounding coins in their stipulated chronological order (commuting only independent tapes as allowed by the protocol). Every natural-error increment has conditional mean zero immediately before its fresh draw, and its conditional second moment is bounded by the applicable envelope in Lemma 19. Multiplying an increment by an indicator decided before that increment preserves conditional centering. In particular, selected direct-source increments and M2R/aggregate-a increments are orthogonal* across stages within one released coefficient*, even when the selected support and the cached rows are statistically dependent.*

Proof. The current support and packing assignments are already measurable at the start of the declared filtration. Local Gaussian errors are fresh, CRT digits are centered given their input residues, and each ModDown remainder is centered given its operand, including all previously sampled digits. For M2R the input A-side is known before its new digits and remainders; the aggregate-a compressor runs only after its input is fixed. Thus each new increment is a martingale difference conditional on the actual cache/history, not merely conditional

on the public parameters. Predictable support indicators preserve this property. Lemma 17 then removes cross terms among these fresh increments. It does *not* assert independence of entire output errors across rounds, or orthogonality after conditioning on future released outputs.

Proposition 2 (Sparse whole-execution arithmetic energy). *Under the above sampling conditions and the protocol's fresh-coin chronology, the squared integer-lift flooding shifts over all released Sparse model coordinates and all auxiliary coordinates admit the bounds*

$$\begin{aligned} \mathcal{E}_m^{F,\mathrm{SP}} &\le TK_\rho B\, W_2^2 V_{\mathrm{loc}}^{\mathrm{ZM}}(n) + \mathbf{1}_{\{k>1\}} L_m V_{\mathrm{M2R}}^{\mathrm{ZM}}(n) + L_m V_{ca}^{\mathrm{ZM}}(\mathrm{model}), \\ \mathcal{E}_a^{F,\mathrm{SP}} &\le L_a W_2^2 V_{\mathrm{loc}}^{\mathrm{ZM}}(n) + L_a V_{ca}^{\mathrm{ZM}}(\mathrm{aux}). \end{aligned} \tag{42}$$

No independence between adaptive selection indicators and cached key material is assumed.

Proof. For a released coordinate r, expand its integer-lift natural error into the contributions of each current direct source, the optional M2R stage, and aggregate-a compression. By Lemma 20 their fresh increments are chronologically centered and orthogonal, even after the setup cache, support, and past output history are exposed. It remains to sum their second moments *without* assuming that cached errors are independent of future support choices.

Let $J_{i,r} \in \{0,1\}$ indicate that source i contributes to released model coordinate r. The support rule gives the pathwise bound $\sum_r J_{i,r} \le TK_\rho B$ for every i; this count includes repeated rounds and does not depend on how outputs influenced later support choices. By the common setup-measurable envelope X_i^{dir} of Lemma 19,

$$\begin{aligned} \mathbb{E} \sum_r J_{i,r}\, \alpha_i^2 \operatorname{Var}(Z_{i,r} \mid \mathcal{F}_{r^-}) &\le \alpha_i^2\, \mathbb{E}\big[X_i^{\mathrm{dir}} \sum_r J_{i,r}\big] \\ &\le TK_\rho B\, \alpha_i^2 \mathbb{E} X_i^{\mathrm{dir}}. \end{aligned}$$

Here $\mathcal{F}_{r^-}$ is the information immediately before the relevant new arithmetic randomness. Crucially, one *single* cached row-norm variable X_i^{dir} dominates every adaptive use by source i. Replacing it by different coordinatewise variables having the same expectation would not justify this incidence step. Sum over i and apply (39) to obtain the direct model term $TK_\rho B W_2^2 V_{\mathrm{loc}}^{\mathrm{ZM}}(n)$.

There are at most L_m released model-coordinate uses, so the same pathwise argument with the *common* cached envelope X^{M2R} gives $L_m V_{\mathrm{M2R}}^{\mathrm{ZM}}(n)$ when $k > 1$; this term is absent when $k = 1$. For aggregate-a rounding, X_{model}^{ca} similarly yields $L_m V_{ca}^{\mathrm{ZM}}(\mathrm{model})$. The auxiliary path has exactly L_a dense coordinate uses, no M2R stage, and consequently has direct and aggregate-a energies bounded by $L_a W_2^2 V_{\mathrm{loc}}^{\mathrm{ZM}}(n)$ and $L_a V_{ca}^{\mathrm{ZM}}(\mathrm{aux})$, respectively. Combining these orthogonal *fresh-increment* energies yields (42). This argument uses expected setup energies but never refreshes a cached key or conditions on future adaptive outputs.

Full deterministic arithmetic energy. Full's deterministic digits and ModDown remainders need not be conditionally centered. Their energy is bounded by Minkowski's inequality rather than by the Sparse martingale decomposition. Set

$$\begin{aligned} A_2 &:= \Big(\sum_i \alpha_i^2\Big)^{1/2}, \\ G_{\mathrm{det}}^2 &:= \frac{N}{P^2}\sum_{\lambda<m}(q_\lambda-1)^2, \\ M_{\mathrm{MD}}(n) &:= \tfrac{1}{2}+\sqrt{Nn/6}, \\ S_{\mathrm{dir}} &:= A_2\sqrt{v_e}+A_2G_{\mathrm{det}}\sqrt{v_e}\sqrt{1+4Nn/3} \\ &\quad +128M_{\mathrm{MD}}(n), \\ R_{\mathrm{nat}} &:= \sqrt{N}\,S_{\mathrm{dir}}, \\ R_{ca}^{\mathrm{F}}(c) &:= E_a(c)N\sqrt{2n/3}. \end{aligned} \tag{43}$$

Proposition 3 (Full arithmetic energy). *Let $R_m = \lceil d/N \rceil$, $R_a = \lceil d_{\mathrm{aux}}/N \rceil$, $R_m^F = R_{\mathrm{nat}} + R_{ca}^{\mathrm{F}}(c_a)$, and $R_a^F = R_{\mathrm{nat}} + R_{ca}^{\mathrm{F}}(c_a^{\mathrm{aux}})$. Under the fixed-cohort Full sampling contract,*

$$\begin{aligned} \sum_{\nu\in\text{Full model}} \mathbb{E}\|\Delta_\nu^{\mathbb{Z}}\|_2^2 &\le TR_m(R_m^F)^2, \\ \sum_{\nu\in\mathrm{aux}} \mathbb{E}\|\Delta_\nu^{\mathbb{Z}}\|_2^2 &\le TR_a(R_a^F)^2. \end{aligned} \tag{44}$$

Proof. With full dense transmission, output positions are not selected by data-dependent support. For a fixed position, expand the encryption/key-switching error using Lemma 5. The local encryption error contributes at most $A_2\sqrt{v_e}$ in L_2. The canonical CRT digits and cached-row errors contribute at most $A_2G_{\mathrm{det}}\sqrt{v_e}\sqrt{1+4Nn/3}$. Each source's centered P-remainder has magnitude at most $P/2$, and the secret-multiplied A remainder contributes at most $\sqrt{Nn/6}$ in coefficientwise L_2; summing fixed weights gives the $128M_{\mathrm{MD}}(n)$ term. Thus the natural-error vector has L_2 norm at most R_{nat}.

Full aggregate-a quantization is deterministic; its integer lift has sup norm at most $E_a(c)$. The collective ternary secret has expected squared norm $2Nn/3$. Bounding the negacyclic product using the independent setup-time secret and the a-side mask distribution gives the $R_{ca}^{\mathrm{F}}(c)$ term. Apply Minkowski's inequality to the natural and aggregate-a errors. Finally, sum the per-instance expectations over the fixed TR_m model and TR_a auxiliary instances. The argument uses linearity of expectation, not independence of reused KSK rows across rounds.

5.8 Gaussian Translation and the H4 Interface

Let P_ν denote the projection onto coefficients that are actually released in instance ν (the identity for Full and auxiliary). A fixed honest pivot $p^\star \in \mathcal{C}\setminus\mathcal{A}$ has

a fresh integer Gaussian smudging share for each instance. The pre-smudging variables, current arithmetic coins, cached keys, and current support are exposed to a filtration *before* the pivot Gaussian is sampled. Therefore $P_\nu \Delta_\nu^{\mathbb{Z}}$ is measurable with respect to that filtration even in adaptive Sparse executions.

Lemma 21 (Integer discrete-Gaussian translation cost). *For $D_{\mathbb{Z},\sigma}$ with probability mass proportional to $\exp(-x^2/(2\sigma^2))$ and every integer shift d, the KL divergence between this distribution and its translate by d is exactly*

$$D_{\mathrm{KL}}\big(D_{\mathbb{Z},\sigma} \,\|\, (D_{\mathbb{Z},\sigma} + d)\big) = \frac{d^2}{2\sigma^2}. \tag{45}$$

The identity also holds for a product of independently sampled coefficients, with the squared shifts summed.

Proof. Integer translation leaves the normalization constant unchanged. Taking the log-ratio of the two masses at x gives $((x-d)^2 - x^2)/(2\sigma^2)$. The expectation of x under the centered discrete Gaussian is zero, leaving $d^2/(2\sigma^2)$. The vector identity follows from additivity of KL divergence for product measures.

Proposition 4 (Chronological flooding cost). *In the H_4 transformation, move only the pivot's released smudging coefficients by $-P_\nu \Delta_\nu^{\mathbb{Z}}$, leaving unreleased coefficients unchanged. If $\mathcal{E}_m^F, \mathcal{E}_a^F$ bound the expected whole-execution squared shifts over the model and auxiliary paths, respectively, then*

$$\Delta_{\mathrm{TV}}(H_3, H_4) \le \epsilon_F := \frac{1}{2}\sqrt{\frac{\mathcal{E}_m^F}{\sigma_{\mathrm{smg}}^2} + \frac{\mathcal{E}_a^F}{(\sigma_{\mathrm{smg}}^{\mathrm{aux}})^2}}. \tag{46}$$

This bound neither assumes independent future supports nor adds a separate setup-good failure probability to privacy advantage. Correctness/no-wrap failure is accounted for separately. Section 6 defines the exact hybrid distributions.

Proof. Enumerate smudging coordinates in chronological order, and condition only on the history immediately preceding the current fresh pivot Gaussian. Its integer shift is already fixed under this conditioning. Applying Lemma 21 at each step and the KL chain rule gives

$$D_{\mathrm{KL}}(H_3 \| H_4) \le \frac{1}{2}\mathbb{E}\left[\sum_{\nu,j \text{ released}} \frac{(P_\nu \Delta_\nu^{\mathbb{Z}}[j])^2}{\sigma_\nu^2}\right].$$

(The inequality allows any additional identically conditioned post-processing.) Pinsker's inequality gives $\Delta_{\mathrm{TV}} \le \sqrt{D_{\mathrm{KL}}/2}$; grouping model and auxiliary contributions yields (46). The use of chronological conditioning is essential: conditioning on future outputs would invalidate the claim that the current shift is determined before the Gaussian is sampled.

5.9 Uniform-Polynomial Samplers and Proof Interfaces

For $\mu < 2^{64}$ prime, the exact UPoly sampler accepts 64-bit words below $L_\mu = \lfloor 2^{64}/\mu \rfloor \mu$, then reduces each accepted word modulo μ.

Lemma 22 (Uniformity and bounded rejection). *In the ideal-XOF model, each accepted residue is exactly uniform on $\mathbb{Z}_\mu$, and successive accepted residues are independent at distinct domain-separated addresses. A one-word acceptance probability is $p_\mu = L_\mu/2^{64} > 1/2$. If at most L words are allowed to generate one residue, the probability that the bound is exhausted is $(1-p_\mu)^L$; for Q_{samp} requested coefficients and at most m_{samp} RNS primes per coefficient, the union bound is at most*

$$\delta_{\mathrm{rej}} \le Q_{\mathrm{samp}} m_{\mathrm{samp}} \max_{\mu} (1-p_\mu)^L. \tag{47}$$

The ideal-XOF programming and prior-query costs are additional *hybrid-game issues; the lemma alone does not establish a ROM transition.*

Proof. Exactly L_μ/μ acceptable 64-bit words map to each residue, so an accepted residue is uniform. Fresh ideal-XOF words give independence. The failure probability after L independent trials is $(1-p_\mu)^L$; take a union bound over the requested outputs. The lower acceptance bound follows from the inequality $\lfloor 2^{64}/\mu \rfloor \mu > 2^{63}$ for $1 < \mu < 2^{64}$.

Interfaces to subsequent sections. The next section formalizes the whole-view H_0–H_9 transitions, including DD and base-XOF programming, implementation-sampler distance, the exact H_4 shift, and the ssRLWE reductions with cached-key consistency. The terminal-game conditioning and the efficient TILT sampler are treated separately in the terminal-simulation section. The present section establishes the algebraic identities and the moments on which those transitions depend; it does not substitute for the terminal-game simulation proof or the numerical parameter certificates.

6 Whole-View Security Proof via Hybrid Games

This section develops the hybrid argument for the most demanding regime, Sparse with $k > 1$. The argument applies to the *entire* joint view of one semi-honest server and a statically corrupted client set, not to an isolated encryption or a single round. The main target is an intermediate *terminal distribution* H_9 in which the honest clients' local ciphertext contents are uniform and the secret-key-dependent natural error of the released decryption phase has been removed by flooding. Constructing an efficient simulator of the H_9 transcript from the permitted leakage is a separate, indispensable step (Section 7, which follows this section).

For transparency, we give the exact order of transformations, the conditional-randomness interfaces, and the reduction costs. The underlying correctness and second-moment statements have been established in Section 5. The Full and

Sparse $k = 1$ versions omit the M2R hybrid but retain the applicable PK, KSK, local ciphertext and decryption steps. Section 8 presents their complete instantiations separately.

6.1 Scope, Distributions, and Chronological Conventions

The joint experiment. Fix a security parameter, a public configuration Cfg, a registered cohort $\mathcal{C}$ of size n, and a corrupted client set $\mathcal{A} \subseteq \mathcal{C}$ of size at most c. Put $\mathcal{H} = \mathcal{C} \setminus \mathcal{A}$ and $h = |\mathcal{H}|$. Throughout this section $h \geq 2$ and $\tau \geq c + 2$; every released Sparse label has at least τ contributors. We consider T completed, fixed-cohort rounds without dropout, survivor revision, or retry. The adversary obtains $\mathsf{View}_{\mathcal{A}}^{\mathrm{real}}$ from (5), including the server's cached state, random tape, all observable messages, the corrupted clients' complete states, and every publicly computed intermediate object. The hybrid distribution is always the *pair*

$$\mathcal{J}_j = (\Lambda_j^{MO}, \mathsf{View}_j), \qquad 0 \leq j \leq 9,$$

where the leakage Λ_j^{MO} is *computed in that game*, not imported from a different game's output law. In particular, a flooding shift is permitted to change the joint distribution of the raw released value O; the prescribed-leakage comparison is dealt with by the conditional-distribution transfer in the terminal-simulation section. We write H_j for $\mathcal{J}_j$ when no ambiguity arises.

Application causality. The update, support selection and application state in round t may be PPT functions of the application randomness and the previously released semantic aggregates and outputs. They cannot depend on the unreleased current cryptographic randomness or on secret past cryptographic transcript bytes. This is the execution contract of Section 3.4, not an independence assumption between successive model updates. Each selected support and each encoded update is fixed before the fresh local a mask and encryption noise used for its ciphertext are revealed.

Caches and observable randomness. All clients' public-key shares, the client-to-collective KSK rows, and, for $k > 1$, the M2R share rows are sampled at setup and are *reused* throughout the execution. Consequently a hybrid replaces a cached row only once, and every subsequent use refers to that same replacement. The public CRS masks are shared exactly where specified by the protocol; they are not sampled independently for each client or each occurrence. Corrupted clients' secret keys, data and stipulated random tapes remain part of the view, and their setup rows are generated consistently under the current public target key. The server's deterministic and randomized post-processing is rerun under the modified cached material.

ssRLWE notation. For $R \in \{R_Q, R_{QP}\}$, let $\epsilon_R(q)$ denote the maximal advantage of a polynomial-time distinguisher between q samples

$$\big(a_j, -a_j s + e_j\big)_{j=1}^{q} \quad \text{and} \quad \big(a_j, U_j\big)_{j=1}^{q}, \tag{48}$$

where a single $s \leftarrow \chi_u$ is reused in one challenge, a_j, U_j are independent uniform elements of R, and $e_j \leftarrow \chi_e$. The convention is unchanged if the secret sign is reversed, since χ_u is symmetric. All reductions below are classical and may use the same challenge public masks for additional independently generated rows. We use Δ_{TV} for statistical total-variation distance, and $\mathrm{Adv}(H, H')$ for the distinguishing advantage of computationally adjacent experiments.

The chronological frontier. For every released model or auxiliary decryption instance ν, let $\mathcal{F}_\nu^{\mathrm{pre}}$ contain the current setup cache, semantic support, local seeds and masks, randomized server arithmetic up to and including the aggregate-a compression, and the error variables that appear before the fresh pivot smudging Gaussian. Crucially, this filtration does not contain future supports, the current pivot Gaussian, or the decryption transcript generated from it. Let P_ν project onto the coefficients actually released; P_ν is the identity on Full and auxiliary blocks and a selector of occupied Sparse lanes on Sparse model blocks. The integer lift $\Delta_\nu^{\mathbb{Z}}$ is defined by Lemma 15, and $P_\nu \Delta_\nu^{\mathbb{Z}}$ is measurable with respect to $\mathcal{F}_\nu^{\mathrm{pre}}$.

Hybrid ledger. The nine transitions are, in order: deferred DD-polynomial sampling; implementation sampler idealization; base-XOF mask lifting; chronological flooding; pivot M2R share uniformization; pivot PK uniformization; removal of honest KSK payloads; restoration of the pivot PK; and removal of honest local ciphertext plaintext dependence. *The ordering of the three computational steps around the KSK transition is essential.* In Sparse $k > 1$, the pivot's M2R share must first be uniformized, so that the PK switch cannot be combined with an unremoved M2R relation involving the pivot threshold secret.

6.2 The Real Experiment and the Statistical Hybrids H_1–H_4

Hybrid H_0: real execution. Game H_0 is exactly the fixed-cohort Sparse $k > 1$ execution from Section 4. The adversary sees the whole transcript, including seed bytes and the corresponding XOF outputs, source ciphertext projections, KSK and M2R caches, labels and their admission metadata, compressed aggregate-a requests, compressed partial decryptions, and final outputs. The joint distribution of the computed leakage and view is

$$H_0 = (\Lambda^{MO}, \mathsf{View}_{\mathcal{A}}^{\mathrm{real}}).$$

No game has altered the meaning of the permitted semantic trace or of M; in particular, M is the encoded weighted sum without cryptographic or compression error.

Hybrid H_1: deferred DD polynomials. After an instance's compressed aggregate-a code is fixed, the real protocol chooses a fresh 128-bit DD root and uses domain-separated ideal-XOF streams to form Off_ν and $D_{i,\nu} \in R_Q$. We replace the future DD polynomial values by independent uniform polynomials in R_Q,

maintaining one entry per specified domain-separated address in a lazy oracle table. The root is still included in the public transcript at the stipulated time. When that root is exposed, the reduction generates the rejection-sampler word streams *conditional on the predetermined residues*; it programs only previously unqueried addresses. For each RNS prime $\mu < 2^{64}$ and target residue d, an accepted word is uniformly chosen from

$$\mathcal{W}_{\mu,d} = \{w < \lfloor 2^{64}/\mu \rfloor \mu : \ w \bmod \mu = d\},$$

while rejected words are sampled from the corresponding rejection range. Each accepted target residue has the same number of such preimages. Hence the target residue, together with the conditionally sampled words, has exactly the original ideal-XOF distribution until a collision or bounded rejection failure occurs.

Let q_{root}^{DD} be the number of fresh DD roots, $q_X^{DD} < 2^{128}$ the adversary's pre-exposure query cap, and L_{DD} the word-attempt cap per coefficient and limb. Write $p_\mu = \lfloor 2^{64}/\mu \rfloor \mu / 2^{64}$, and let $N_{DD,\mu}$ be an upper bound on the total number of coefficient requests for prime μ. A conservative error ledger is

$$\delta_{\mathrm{DD-ROM}} \le \frac{q_{\mathrm{root}}^{DD} q_X^{DD}}{2^{128} - q_X^{DD}} + \sum_{\mu | Q} N_{DD,\mu} (1 - p_\mu)^{L_{DD}}. \tag{49}$$

For the single-successful-attempt-per-round contract, $q_{\mathrm{root}}^{DD} = T$; a whole-polynomial conservative count is $N_{DD,\mu} \le NT(R_{\max} + R_a)(n+1)$, where $R_{\max}$ bounds model decryption groups per round and $R_a = \lceil d_{\mathrm{aux}}/N \rceil$. The union bound over roots and prime limbs yields

$$\Delta_{\mathrm{TV}}(H_0, H_1) \le \delta_{\mathrm{DD-ROM}}.$$

The *third* DD programming cost needed after conditioned suffix sampling is distinct: it belongs to Section 7 and is not yet charged here. The DD root is never retroactively programmed after a previously exposed address has been queried.

Hybrid H_2: implementation samplers. Game H_2 changes the prescribed finite-precision sampler implementations to the ideal exact distributions used in the proofs. For sampler class r, suppose its conditional total-variation distance from the ideal sampler is at most η_r for every allowed prior transcript, and it is called at most N_r times during the entire execution. We couple each call in temporal order, conditional on an identical preceding transcript. On the first disagreement the coupling may diverge; a union bound over the remaining conditional experiments yields

$$\Delta_{\mathrm{TV}}(H_1, H_2) \le \delta_{\mathrm{impl}} := \sum_r N_r \eta_r. \tag{50}$$

This step is not a claim that a real machine implements an ideal sampler for free. Where a sampler is exact, $\eta_r = 0$; otherwise a per-class certificate and

a global invocation bound are required. The corrupted clients' randomness is retained in the joint experiment, so a sampler replacement is performed only on the stipulated honest or public sampling calls and its induced conditional view distribution.

Hybrid H_3: base-XOF mask lifting. The common setup mask p_1, the km M2R common masks $a^M_{j,\lambda}$, and each honest local encryption mask $a_{i,t,r}$ are generated from distinct domain-separated 128-bit seed/address families. Game H_3 represents them first as independent uniform ring elements, and then programs the still-unread XOF word streams so that the publicly revealed seeds reconstruct exactly those elements. These are *one-time* associations: all future calls involving a cached mask use the same ring element and oracle table entry. For an honest client's new upload, its local mask is associated with the fresh seed *before* that seed's first exposure. Corrupted participants' supplied material is left as in the real experiment.

For Sparse, define

$$R_{\mathrm{loc}} = \lceil K_\rho/k \rceil, \qquad R_a = \lceil d_{\mathrm{aux}}/N \rceil, \qquad \kappa_k = \mathbf{1}_{\{k>1\}}.$$

Conservative seed-root and coefficient-programming counts are:

$$\begin{aligned} q^{base}_{\mathrm{root}} &\le 1 + nT(R_{\mathrm{loc}} + R_a), \\ C_{\mathrm{base}} &\le N\big[(1+\kappa_k km)\ell_{QP} + nT(R_{\mathrm{loc}} + R_a)\ell_Q\big], \end{aligned} \tag{51}$$

where ℓ_Q and ℓ_{QP} denote the numbers of RNS primes in the corresponding rings. With pre-exposure query cap $q^{base}_X < 2^{128}$ and finite rejection cap L_{base}, set

$$\begin{aligned} \delta_{\mathrm{base-ROM}} \le{}& \frac{q^{base}_{\mathrm{root}} q^{base}_X}{2^{128} - q^{base}_X} \\ &+ N(1+\kappa_k km) \sum_{\mu \mid QP} (1-p_\mu)^{L_{base}} \\ &+ nNT(R_{\mathrm{loc}} + R_a) \sum_{\mu \mid Q} (1-p_\mu)^{L_{base}}. \end{aligned} \tag{52}$$

Each term follows by exact inverse-rejection sampling as in H_1 and Lemma 22, plus a union bound over first-query collisions. The coupling preserves the joint CRS, local seeds, cached keys, and observable oracle queries, not merely their marginal masks:

$$\Delta_{\mathrm{TV}}(H_2, H_3) \le \delta_{\mathrm{base-ROM}}.$$

The base-XOF table is part of the setup/frontier carried to the terminal game. It will *not* be reprogrammed again after the terminal-game suffix is sampled; this restriction is necessary to avoid silently changing previously exposed seeds.

Hybrid H_4: chronological Gaussian flooding. Choose a fixed honest pivot $p^\star \in \mathcal{H}$. For each model or auxiliary decryption instance ν, let T_ν be the pre-c_2 released

phase, and let M_ν be its exact encoded aggregate. Lemmas 13 and 15 imply, on released coefficients,

$$P_\nu(T_\nu - M_\nu) = P_\nu U_{1,\nu} + P_\nu G_\nu + P_\nu \Delta_\nu^{\mathbb{Z}} \pmod{Q}, \tag{53}$$

where $U_{1,\nu}$ denotes the transported c_1 error, $G_\nu = \sum_{i\in\mathcal{C}} g_{i,\nu}$ is the total fresh smudging noise, and $\Delta_\nu^{\mathbb{Z}}$ contains the natural encryption, KSwitch, M2R, and aggregate-a compression errors but excludes $U_{1,\nu}$ and G_ν. The lift must be taken in $\mathbb{Z}^N$ consistently with the actual arithmetic; taking a centered representative of an arbitrary modular expression is not a substitute.

The only distributional change from H_3 is that the pivot's *fresh* smudging vector is replaced by

$$g_{p^\star,\nu} \;\longmapsto\; g_{p^\star,\nu} - P_\nu \Delta_\nu^{\mathbb{Z}} \qquad \text{on released coefficients.} \tag{54}$$

Unreleased coordinates are unchanged. All downstream response codes, server reconstructions, and released outputs are recomputed from the shifted value; no transcript entry remains from the inconsistent unshifted execution. Equation (53) now becomes

$$P_\nu(T_\nu^{H_4} - M_\nu) = P_\nu(U_{1,\nu} + G_\nu^{\text{unshifted}}) \pmod{Q}. \tag{55}$$

This removes the source- and target-secret-dependent *natural* error from the released phase, but it does *not* remove the U_1 term or turn individual partial-decryption messages into independent samples. Both effects are handled in the terminal conditional-law analysis.

Lemma 23 (Whole-execution cost of H_4). *Let $\mathcal{E}_m^F$ and $\mathcal{E}_a^F$ be valid expected squared integer-shift bounds, with the first restricted to released model coefficients and the second to the auxiliary path, as in Propositions 2 and 3. If the pivot smudging widths are σ_{smg} and $\sigma_{\text{smg}}^{\text{aux}}$, then*

$$\Delta_{\text{TV}}(H_3, H_4) \le \epsilon_F := \frac{1}{2}\sqrt{\frac{\mathcal{E}_m^F}{\sigma_{\text{smg}}^2} + \frac{\mathcal{E}_a^F}{(\sigma_{\text{smg}}^{\text{aux}})^2}}. \tag{56}$$

Proof. At each chronological instance, condition on the full past and the current $\mathcal{F}_\nu^{\text{pre}}$ but not the fresh pivot Gaussian. The shift in (54) is then a measurable integer vector. Lemma 21 gives conditional KL cost $\|P_\nu\Delta_\nu^{\mathbb{Z}}\|_2^2/(2\sigma_\nu^2)$. Sum the costs chronologically by the KL chain rule: previously released outputs may affect later supports, but are part of the preceding filtration. By data processing, appending the deterministic and PPT response-generation and output steps cannot increase KL. Pinsker's inequality gives (56). The required whole-execution energy bounds are those of Section 5.7; none treats adaptive incidence variables as independent of cached switching-key rows.

Lemma 24 (Pivot response from the flooded phase). *Fix an instance ν of the H_4 experiment, the full public server-side arithmetic through the aggregate*

ciphertext, the $U_{1,\nu}$ transport error, the semantic aggregate M_ν, and all nonpivot partial-decryption responses. If $g_{p^\star,\nu}$ is freshly sampled with its stipulated smudging law, the pivot's post-shift *raw response on released coordinates may be generated without the pivot's threshold secret by*

$$P_\nu h^{H_4}_{p^\star,\nu} = P_\nu\big(M_\nu + U_{1,\nu} + G_{\neg p^\star,\nu} + g_{p^\star,\nu} - b_\nu - \sum_{i\neq p^\star} h_{i,\nu}\big) \pmod Q, \tag{57}$$

where $G_{\neg p^\star,\nu} = \sum_{i\neq p^\star} g_{i,\nu}$. This identity is a construction of the flooded raw response*. It does not assert that a collection of response codes drawn independently from their marginals has the correct joint law.*

Proof. Before the H_4 shift, write the pivot's raw response as $h_{p^\star,\nu} = \widehat{a}_\nu \xi_{p^\star,\nu} + g_{p^\star,\nu}$ and the other responses according to the same protocol. The exact arithmetic-phase identity is $T_\nu = M_\nu + U_{1,\nu} + G_\nu + \Delta^{\mathbb{Z}}_\nu$ modulo Q. Replacing only the pivot smudging by $g_{p^\star,\nu} - P_\nu \Delta^{\mathbb{Z}}_\nu$ changes its response by that same shift, and gives $b_\nu + \sum_i h^{H_4}_{i,\nu} = M_\nu + U_{1,\nu} + G_\nu$ on released positions. Solving for the pivot response yields (57). Everything on its right-hand side is sampled or computed by a full-experiment reduction, which knows the application inputs, current semantic aggregate, and nonpivot keys. The law is identical to the post-shift response law because it is obtained by an algebraic rearrangement of the same underlying random variables. Compression of the resulting response uses the protocol's fresh quantizer randomness and therefore preserves its correlation with the other responses. The additional *output-conditioned* generation required of a leakage-only simulator is a separate issue for Section 7.

6.3 The Computational Hybrids H_5–H_9

Computational substitutions start only after H_4. The point of this order is that the encrypted aggregate's observable decryption *total* is no longer forced to retain the pivot secret through a natural-error term. The following reductions use the joint server–colluder experiment, including consistent response-generation kernels for the flooded phase. A simulator for the *prescribed* output-conditioned version of those kernels is constructed and validated in Section 7. The computational hybrids themselves must not assume that the original key-dependent responses remain unchanged after a secret is removed.

Reduction interfaces and one-cache replay. Fix the pivot $p^\star \in \mathcal{H}$. The challenge is inserted into a *single* setup and its subsequent T rounds. The relevant cached families are

$$\begin{aligned} \mathcal{K}^M_{p^\star} &= \{(a^M_{j,\lambda}, b^M_{j,\lambda,p^\star}) : j < k, \lambda < m\}, \\ \mathcal{K}^{KS}_{\mathcal{H}} &= \{(A_{i,\lambda}, B_{i,\lambda}) : i \in \mathcal{H}, \lambda < m\}, \\ \mathcal{P}_{PK} &= (p_1, p_{0,\mathcal{C}}), \qquad p_{0,\mathcal{C}} = \sum_{i\in\mathcal{C}} p_{0,i}. \end{aligned} \tag{58}$$

Here $\mathcal{K}^M_{p^\star}$ exists only when $k > 1$; $a^M_{j,\lambda}$ is the common mask used by *all* client M2R shares. Similarly, every $A_{i,\lambda}$ in $\mathcal{K}^{KS}_{\mathcal{H}}$ uses the shared public p_1. An upload $b_{i,j}$ is observable only through its specified coefficient projection $P_{S_{i,j}} b_{i,j}$ (and compression).

Table 2 gives, for *each* edge, the secret hidden from its reduction, the challenge insertion, and the exact types of dependent data that must be recomputed. The symbols Z, Z_1, Z_2 denote the appropriate component(s) of an ssRLWE challenge (48); in the genuine branch these are noisy secret multiples, and in the uniform branch they are uniform ring elements. Formulas appear in full in the corresponding reduction proofs below.

Table 2. Concrete interfaces for the $H_4 \to H_9$ reductions. A challenge substitutes one cached row or one chronological source family; the other rows use the same public masks.

Edge	Hidden key and challenge substitution	Recomputed dependencies
$4 \to 5$	Unknown $t_{p^\star}$, known $s_{p^\star}$: $b^M_{j,\lambda,p^\star} \leftarrow Z_{j,\lambda} + P\gamma_\lambda \Psi_j(s_{p^\star})$	Cohort M2R cache, repacking, and flooded responses
$5 \to 6$	Unknown $s_{p^\star}$: $(p_1, p_{0,p^\star}) \leftarrow (a, Z)$	Collective target $p_{0,\mathcal{C}}$ and every target-bound KSK row
$6 \to 7$	Unknown $u_{i,\lambda}$, known x_i: $(A_{i,\lambda}, B_{i,\lambda}) \leftarrow (Z_1, Z_2 + P\gamma_\lambda x_i)$	All uses of that honest KSK row, without new public masks
$7 \to 8$	Unknown $s_{p^\star}$: reverse the (a, Z) PK challenge; honest KSK payloads stay zero	Restored collective target and its zero-payload KSK rows
$8 \to 9$	Unknown x_i, known current $v_{i,j}, S_{i,j}$: $P_{S_{i,j}} b_{i,j} \leftarrow P_{S_{i,j}}(Z_j + v_{i,j})$	Compressed uploads, switching, aggregation, and flooded responses

All reductions *know* the secrets of nonchallenged honest clients and retain the corrupted clients' stipulated keys and coins. Thus, for example, the H5 reduction can form the known payload $P\gamma_\lambda \Psi_j(s_{p^\star})$ without learning $t_{p^\star}$, whereas the H9 reduction forms $P_{S_{i,j}}(Z_j + v_{i,j})$ without learning x_i. For the PK edges, the collective target is updated by (59); it is never kept fixed after replacing $p_{0,p^\star}$. Each changed cache row is created *once* and read unchanged on later rounds. Server-side KSwitch, R2M, label aggregation and M2R are recomputed using the already sampled randomized-arithmetic coins; the response is constructed using Lemma 24 wherever the challenge secret is unknown. Neither the original pivot secret nor a fresh independent copy of a reused cache is needed.

Hybrid H_5: uniformizing the pivot M2R share. For $k > 1$, the setup supplies km M2R rows under independent aggregate-decryption secrets t_i. The pivot's rows

are

$$b^M_{j,\lambda,p^\star} = -a^M_{j,\lambda} t_{p^\star} + P\gamma_\lambda \Psi_j(s_{p^\star}) + e^M_{j,\lambda,p^\star} \pmod{QP}, \qquad j < k,\ \lambda < m.$$

Game H_5 replaces *the same cached km pivot b-shares* with independent uniform polynomials of R_{QP}, retaining the shared public masks $a^M_{j,\lambda}$ and every nonpivot share. The corresponding cohort-summed M2R keys and all server repacking operations are recomputed from these substituted shares.

Lemma 25 (Pivot M2R-share reduction). *Under the post-H_4 generation contract and ssRLWE over R_{QP},*

$$\mathrm{Adv}(H_4, H_5) \le \epsilon_{QP}(km).$$

The cost is incurred once for the setup cache, not once per use.

Proof. A reduction receives km challenge pairs $(a^M_{j,\lambda}, Z_{j,\lambda})$ sharing an unknown small secret $t_{p^\star}$. It samples the pivot's $x_{p^\star}$ and $s_{p^\star}$ (these are independent of $t_{p^\star}$), and all nonpivot honest secrets, while keeping corrupted-client states as specified. It sets the pivot's share to

$$b^M_{j,\lambda,p^\star} = Z_{j,\lambda} + P\gamma_\lambda \Psi_j(s_{p^\star}) \pmod{QP}.$$

Genuine ssRLWE samples give the original cached M2R rows; uniform samples give independent uniform pivot rows because translation by known payloads preserves uniformity. Other clients' shares are computed under *the same* public masks. All derived M2R keys, R2M/M2R server intermediates and transcript fields are generated jointly from the challenge cache. The flooded pivot response is generated by the exact algebraic formula in Lemma 24; this is possible without access to $t_{p^\star}$ and retains correlations among all response codes. Section 7 gives the stronger leakage-only, output-conditioned response sampler. Since the cache is never resampled on later use, this is one vector-ssRLWE challenge with km samples, giving the stated bound.

Hybrid H_6: uniformizing the pivot PK share. Write

$$p_{0,p^\star} = -p_1 s_{p^\star} + f_{p^\star} \pmod{QP}, \qquad p_{0,\mathcal{C}} = \sum_i p_{0,i}.$$

Game H_6 replaces the pivot public-key share with an independent uniform polynomial. This makes the aggregate target $p_{0,\mathcal{C}}$ uniform and independent of p_1; it does not make each other participant's PK share independent of the same p_1. If the pivot share changes from $p_{0,p^\star}$ to $p'_{0,p^\star}$ while all other PK shares are held fixed, the public target used by *every* KSK row changes by

$$p'_{0,\mathcal{C}} = p_{0,\mathcal{C}} - p_{0,p^\star} + p'_{0,p^\star} \pmod{QP}. \tag{59}$$

Every KSK row, including rows held or generated by corrupted clients, is consistently recomputed under this $p'_{0,\mathcal{C}}$, and server key switching and aggregation are rerun. Thus replacing a PK share without updating the KSK target would *not* be the game defined here.

Lemma 26 (Joint PK/target-KSK transition). *Assuming the pivot's M2R rows are already uniform for $k > 1$,*

$$\mathrm{Adv}(H_5, H_6) \le \epsilon_{QP}(1).$$

Proof. Embed one ssRLWE pair (a, Z) as $(p_1, p_{0,p^\star})$. In the genuine branch it is the pivot public-key share; in the uniform branch it is an independent uniform share. For every other honest client the reduction samples the actual s_i, x_i and KSK randomness. It regenerates the corresponding public-key and target-dependent KSK cache from the challenge target $p_{0,\mathcal{C}}$, rather than keeping inconsistent cached rows from the previous game. Before payload removal, the reduction knows all honest local keys x_i and can form their KSK payloads. Corrupted KSK rows are generated from their native fixed secret keys and prescribed randomness against the same target; one cannot replace these rows by simulated honest rows. Publicly computable switching and server states are then recomputed. The independent pivot M2R replacement prevents an additional relation to $s_{p^\star}$ from surviving through that cache. The post-flooded pivot response is generated using Lemma 24. All deterministic shared-mask and reuse correlations are therefore present in both branches, and a distinguisher separates the two branches only by distinguishing the underlying ssRLWE pair.

Hybrid H_7: removal of honest KSK payloads. In H_6 the two public components p_1 and $p_{0,\mathcal{C}}$ are uniform and independent. A target-bound honest row has the form

$$A_{i,\lambda} = u_{i,\lambda} p_1 + e^1_{i,\lambda}, \qquad B_{i,\lambda} = u_{i,\lambda} p_{0,\mathcal{C}} + e^0_{i,\lambda} + P\gamma_\lambda x_i.$$

Order the hm rows of honest clients once and for all. We successively change their B-components by removing the $P\gamma_\lambda x_i$ payload, leaving their A-components and all corrupted rows in their corresponding native distributions.

Lemma 27 (Two-sample reduction for one cached KSK row). *In the public-mask distribution of H_6, the distinguishing advantage between an honest row with payload $P\gamma_\lambda x_i$ and the same row with zero payload is at most $2\epsilon_{QP}(2)$, where the two ssRLWE samples use a common small secret $u_{i,\lambda}$. The statement includes rows sharing the public masks and all their derived server computations.*

Proof. Let $m_i = P\gamma_\lambda x_i$, known to the reduction at this stage. One two-sample ssRLWE challenge produces

$$(A, B) = (Z_1, Z_2 + m_i)$$

with corresponding shared uniform public masks $(p_1, p_{0,\mathcal{C}})$. In the genuine branch, (Z_1, Z_2) has the required pair of noisy multiples of the *same* ternary $u_{i,\lambda}$ (using the sign-symmetric convention). In the uniform branch, $(Z_1, Z_2 + m_i)$ is a uniform pair and translation by the known payload changes nothing. A reverse two-sample experiment gives (Z_1, Z_2) distributed as a zero-payload row. The two paths cost at most $\epsilon_{QP}(2)$ each.

To embed the two challenged public masks without breaking the public-key relation, generate the nonpivot PK shares first and set the pivot public-key share in this H_6 game to $p_{0,p^\star} = p_{0,\mathcal{C}} - \sum_{i \neq p^\star} p_{0,i}$. Because the pivot share is uniform in H_6, this has exactly the correct joint law. All other rows are generated with fresh independent multipliers and errors but with the *same two shared public masks*. The reduction does not claim that rows using those masks are independent after the masks themselves are fixed; it includes their full joint view by generating them and every server-side object with the challenge row in place. A modified row is cached once and reused on every round. The reduction includes the public server randomness and its post-processing of that cached row, so the advantage bound covers the full transcript, not just the KSK pair.

Lemma 28 (All honest KSK rows). *The total replacement cost of hm honest rows satisfies*

$$\mathrm{Adv}(H_6, H_7) \leq 2hm\, \epsilon_{QP}(2).$$

Proof. Apply Lemma 27 to each honest row in a fixed order. After a row is converted its zero-payload cache remains in place, while unconverted rows retain their actual payloads; every intermediate experiment uses the same public masks and consistently recomputed server states. The triangle inequality over hm rows proves the claim.

Hybrid H_8: restoring the pivot PK law. Game H_8 restores $p_{0,p^\star} = -p_1 s_{p^\star} + f_{p^\star}$ while retaining the *zero-payload* KSK rows for honest clients. The pivot M2R share stays uniformized when $k > 1$. All target-bound rows are regenerated under the restored target PK; corrupted rows continue to contain their actual payloads.

Lemma 29 (Reverse PK transition).

$$\mathrm{Adv}(H_7, H_8) \leq \epsilon_{QP}(1).$$

Proof. Use the single-sample PK challenge of Lemma 26 in reverse. For honest clients, a zero-payload KSK row is generated from public target masks and newly sampled u, e without knowledge of the client's local key. Corrupted rows are generated with the corrupted client's actual secret and prescribed random tape. Every cached row is regenerated under the changed target, and all later server transformations are recalculated consistently. The genuine branch restores the original pivot PK distribution; the uniform branch is H_7. The earlier M2R uniformization remains fixed, so there is no hidden reintroduction of the pivot M2R dependency.

Hybrid H_9: removing honest plaintext dependencies. In H_8, every honest local KSK row has zero payload, so no honest local key x_i enters that cache. The remaining occurrences of x_i in the joint execution are the locally encrypted model and auxiliary ciphertext b-components. In particular, the H9 substitution is *not* a substitution of the public plaintext aggregate M_ν: M_ν remains

the exact weighted sum defined in the security experiment, whereas the uploaded b-polynomial becomes computationally uniform. For an honest client and chronologically enumerated upload j,

$$b_{i,j} = P_{S_{i,j}}(-a_{i,j}x_i + e_{i,j} + v_{i,j}), \qquad e_{i,j} \leftarrow \chi_e,$$

where $P_{S_{i,j}}$ is the transmitted-coefficient projection (the identity for Full and auxiliary uploads). The selection $S_{i,j}$ and plaintext $v_{i,j}$ are determined by the allowed semantic history before the fresh current upload encryption randomness.

Lemma 30 (Chronological local-ciphertext reduction). *Let q_x bound the number of local RLWE encryption rows of a single honest client in the fixed-cohort execution. Under the stated application-causality contract, changing all its uploaded b values to the corresponding projections of independent uniform ring polynomials costs at most $\epsilon_Q(q_x)$ in distinguishing advantage. For Sparse $k > 1$, one may take*

$$q_x = T\big(\lceil K_\rho/k \rceil + R_a\big).$$

Proof. The reduction embeds a q_x-sample ssRLWE challenge (a_j, Z_j) for the single unknown honest source key x_i. At step j, the application supplies the support and plaintext before revealing the current challenge value. The reduction forms $P_{S_{i,j}}(Z_j + v_{i,j})$ and inserts it at the correct upload position, with the stipulated seed/mask association and c_1 compression. In the genuine branch this is the original ciphertext distribution. In the uniform branch, the translation $Z_j \mapsto Z_j + v_{i,j}$ preserves uniformity; projection yields the correct uniform marginal on the transmitted coefficients. The post-H_4 pivot response is regenerated by Lemma 24 consistently with that branch; inputs cannot be functions of unseen current challenge components. Unused padded challenge rows are ignored.

Proposition 5 (Computational terminal-game bound). *For Sparse $k > 1$, the cumulative computational cost is*

$$\begin{aligned}\epsilon_{\mathrm{comp}}^{\mathrm{SP,k>1}}(h) := \epsilon_{QP}(km) + 2\epsilon_{QP}(1) \\ + 2hm\epsilon_{QP}(2) + h\epsilon_Q(q_x).\end{aligned} \tag{60}$$

For Sparse $k = 1$, omit the M2R term. For Full, omit the M2R term and use its own (dense) local-row count $q_x = TR_\Sigma$; other profile-specific distinctions in the statistical step are proved in Section 8.

Proof. Sum the costs of Lemmas 25, 26, 28, 29, and 30 over the h honest local-encryption sources. An M2R cache is uniformized once, so no T multiplier appears on that term. A conservative static bound replaces h by n, since all terms are nondecreasing in h.

6.4 Hybrid Accounting and the Terminal Distribution

Theorem 1 (Whole-view distance to the computational terminal game). *For the fixed-cohort Sparse $k > 1$ experiment, under the sampling, bounded-oracle and ssRLWE assumptions stated above, the joint law of the permitted leakage and adversarial view satisfies*

$$\begin{aligned} \mathrm{Adv}(H_0, H_9) &\le \epsilon_{\mathrm{pre}}^{\mathrm{SP,k>1}}(h), \\ \epsilon_{\mathrm{pre}}^{\mathrm{SP,k>1}}(h) &:= \delta_{\mathrm{DD-ROM}} + \delta_{\mathrm{impl}} + \delta_{\mathrm{base-ROM}} \\ &\quad + \epsilon_F + \epsilon_{\mathrm{comp}}^{\mathrm{SP,k>1}}(h). \end{aligned} \tag{61}$$

Here the flooding energy ϵ_F is instantiated by the Sparse model and auxiliary bounds from Proposition 2. Equation (61) *is a* joint-distribution *bound; it is not yet the final prescribed-leakage simulation theorem.*

Proof. Use the sequence of experiments from H_0 through H_9 and the triangle inequality for distinguishing advantages. The first four steps are controlled by (49), (50), (52), and Lemma 23. The remaining five steps are controlled by Proposition 5. All comparisons concern the same full server–colluder view, with the leakage computed inside each experiment, so no claim of unjustified pointwise conditioning is used.

What is, and is not, independent in H_9. The honest *transmitted* b-values are uniform after the final hybrid. This does not make their compressed code transcripts independent of the prescribed outputs, nor does it make local Sparse supports independent of earlier released model values. The cached KSK rows remain shared across rounds, and in the $k > 1$ regime the pivot's M2R share remains uniformized; the nonpivot M2R rows retain their prescribed dependencies. Moreover, the server's collective-decryption transcript is a conditioned code-sum object, not an independent Gaussian sample.

Let SetupSt contain the terminal cached state, public masks, and the corrupted clients' specified state. For each released instance ν, the *frontier* contains the semantic layout, base seeds and a-side quantities fixed before the honest c_1 codes are drawn. The *suffix* contains the honest c_1 codes, the derived b-side processing, fresh smudging and DD variables, c_2 codes and released raw output. In particular, a frontier must not be defined retrospectively to include the honest c_1 transcript or aggregate-b value on which the output conditional law depends.

Section 7 establishes that the terminal joint law admits a chronological factorization compatible with the permitted (M, O) and the semantic trace, and gives an explicit polynomial-time conditional suffix kernel. The desired whole-view simulator must also recover the byte-compatible DD streams from that suffix and satisfy (8) *exactly*. Those obligations are *not* consequences of the ssRLWE substitutions alone.

Roadmap for the remaining security proof. The next section distinguishes Full fixed-block frontiers from Sparse support-conditioned frontiers, handles variable

contributor sets and corrupted c_2 randomness, certifies the runtime of TILT, and restores DD byte streams. It then transfers the conditional sampler to the actual leakage law and derives the final bound. Section 8 states the regime-specific theorems.

7 Terminal-Game Simulation and Conditional Sampling

The hybrids of Section 6 reach a computational terminal distribution H_9, but this alone does not establish the simulation statement (7). In particular, the compressed partial-decryption codes cannot be sampled independently of the prescribed raw output O. This section constructs an efficient *conditional* generator for the terminal transcript. It follows the fixed-cohort specification of Sections 3 and 4 and the sampling and error conventions of Section 5. The most general case here is Sparse with $k > 1$; the same conditional generator also covers Full and Sparse with $k = 1$ after removing inapplicable conversion operations. The profile-specific theorem statements are reserved for the subsequent profile-specific section.

The construction has five components. First, we isolate a *pre-c_1 frontier* whose distribution does not change when the terminal output is prescribed. Second, we give an exact finite-error representation of the normalized c_1/c_2 codes. Third, an exponentially tilted rejection sampler generates these errors conditioned on the output. Fourth, the Gaussian secret shares, response codes and public distributed dithers are reconstructed consistently. Finally, we compose these per-instance kernels in chronological order and transfer the joint-law hybrid estimate to the desired prescribed-leakage experiment. Every statement about efficient sampling below is *conditional on explicit, uniform polynomial-time certificates*; it is not inferred merely from an observed average acceptance rate.

7.1 The Terminal State, Frontier, and Suffix

Write $\mathsf{SetupSt}$ for the single setup/cache object in H_9. It contains the simulated long-term secrets needed to execute nonpivot operations, the joint public key and cache, all target-bound zero-payload honest KSK rows, and, when $k > 1$, the independent uniform pivot M2R-share vector together with nonpivot M2R shares. Every cache entry is sampled *once* and reused in all later rounds. The corrupted clients' state is the supplied $\mathsf{CorSt}_{\mathcal{A}}$; the simulator does not receive the real honest clients' keys or individual updates. Uniform terminal honest upload polynomials also retain their prescribed correlation within a packed ciphertext and across every deterministic server use.

For one released model or auxiliary instance ν, its semantic layout A_ν includes the relevant public profile, contributor sets, admitted shuffled labels, upload supports, packing-lane assignments, aggregation weights, and output positions. Let $\mathsf{Frontier}_\nu$ be the information generated before the honest c_1 quantization suffix: the shared cache, the semantic layout, local public seeds and derived uniform

a-polynomials, the cached KSK A-components, the randomized HYBRID digits and their A-side ModDown coins, the R2M/M2R A-side processing if applicable, the aggregate a_ν, and the aggregate-a compressor's output, randomness and reconstruction $\widehat{a}_\nu$. The random arithmetic coins belong to the frontier *only if they can be generated from this A-side information without inspecting the current honest b-components.* In particular, an uploaded honest c_1 code, $\widehat{b}$, the aggregate b, the response codes and the delayed DD polynomials are *suffix*, not frontier, objects.

We work first in the deferred-polynomial ideal-XOF terminal game of Hybrid H_1. Thus DD values may be treated as independent uniform ring polynomials until their prescribed public seed and byte streams are reconstructed. The conversion back to the byte-level view is performed only in Subsection 7.7.

Lemma 31 (Data-free pre-c_1 frontier). *Fix the setup state* $\mathsf{SetupSt}$*, the permitted layout* A_ν*, the prior history* $\mathsf{Hist}_{<\nu}$*, and the fresh base-mask seeds and public identifiers* B_ν*. In* H_9 *there is a PPT kernel* μ_ν *such that*

$$\mathcal{L}(\mathsf{Frontier}_\nu \mid \mathsf{SetupSt}, A_\nu, \mathsf{Hist}_{<\nu}, B_\nu) = \mu_\nu(\cdot \mid \mathsf{SetupSt}, A_\nu, \mathsf{Hist}_{<\nu}, B_\nu).$$

No current honest model value is read by this kernel. This equality concerns the frontier's generation kernel*; it does not assert that the semantic layout is independent of earlier released outputs.*

Proof. At the terminal game, each new local a is generated from its seed and public address, without a model update. HYBRID decomposition on the A-side uses that a, cached A-rows and fresh randomized arithmetic coins. If $k > 1$, R2M splits those A-components and M2R uses the resulting columns with cached M2R A-rows; all label-wise assignments have already been fixed in A_ν. The aggregate a is therefore an A-side function. Its compressed code is drawn only after a has been fixed, with fresh coins. For $k = 1$ and Full the corresponding conversion steps are absent. The application-causality contract prevents a future semantic choice from reading the cryptographic coins of the current instance. These observations give the claimed conditional kernel by induction over the server operations. The only departures from this ideal polynomial kernel are the base-mask XOF exceptional events already charged in Hybrid H_3; they are not charged again here.

Output-normal form. Set $K_1 = 2^{c_1}$ and $K_2 = 2^{c_2}$, with path-specific values on model and auxiliary blocks. In the Sparse model path let $I_H(\nu)$ denote honest contributors to this particular released chunk/label; on Full or auxiliary paths put $I_H(\nu) = \mathcal{H}$. Every member of $\mathcal{H} = \mathcal{C} \setminus \mathcal{A}$ participates in collective decryption, including honest clients that did not upload the current Sparse chunk. Thus I_H affects the c_1 error but *not* the number $h = |\mathcal{H}|$ of honest smudging and c_2 contributors.

Lemma 32 (Terminal output identity). *For a released coefficient* ν *in the flooded terminal game, there are an integer* $k^{\mathrm{out}}_{\mathcal{A},\nu}$ *determined by permitted corrupted-client state and semantic context, a finite honest-upload error* $U^H_{1,\nu}$*,*

a finite centered response-code error $U_{2,\nu}$, and a sum of fresh honest smudging integers $G_{\Sigma,\nu}$ such that

$$O_\nu = M_\nu + k^{\mathrm{out}}_{\mathcal{A},\nu} + U^H_{1,\nu} + U_{2,\nu} + G_{\Sigma,\nu} \pmod{Q}. \tag{62}$$

Conditioned on the permitted context, the law of the right-hand finite-error and smudging variables depends on I_H, the public weights and the number of honest decryption clients, but not on the numerical values of $\mathsf{SetupSt}$ *or the pre-c_1 frontier.*

Proof. Expand the actual code-sum reconstruction from (18), separate the terms attributable to corrupted users, and use the exact flooding cancellation of Lemma 24. The flooded natural key-switching, encryption, M2R and compressed aggregate-a errors cancel from the output sum on released positions. The honest upload codes leave U^H_1, the single sum-before-reconstruction c_2 quantizer leaves U_2, and the unswitched honest smudging shares leave G_Σ. The corrupted shift, including the fixed-coin centering correction defined below, is known from the prescribed corrupted state. In H_9 the honest transmitted b-projections are uniform, and deferred DD randomizes response-code inputs. Hence the finite primitive laws derived below are independent of the frontier. This independence is stated for the terminal reference law, not for unmodified real-world ciphertexts.

Lemma 33 (Terminal posterior sufficiency). *For any permitted context $D = (M_\nu, A_\nu, \mathsf{CorSt}_{\mathcal{A}})$ in the support of H_9,*

$$\Pr_{H_9}[O_\nu = o \mid D, \mathsf{SetupSt}, \mathsf{Frontier}_\nu] = \Pr_{H_9}[O_\nu = o \mid D].$$

Consequently, conditioned on D and $O_\nu = o$, the setup/frontier law agrees with its law conditioned on D alone. Furthermore, given the permitted shuffled-label layout, the terminal cryptographic *kernel does not use the inverse secret permutation K_{perm}; no hidden original-index key is required to sample the server–colluder cryptographic view.*

Proof. Equation (62) expresses the output likelihood through variables having a fixed finite/Gaussian law once D is specified. This likelihood is constant as a function of the cache and frontier, so Bayes' rule cancels it from numerator and denominator. For the final assertion, after the H9 removal of honest plaintext payloads, the server's downstream operations use only the observed shuffled labels, contributor bitmaps and lane/group assignments. The inverse mapping to original model indices is needed by the client-side application, not by this terminal cryptographic kernel. The semantic application tape itself is not thereby claimed to be independent of K_{perm}.

7.2 Exact Finite Quantization Primitives

We adopt the single deterministic nearest-integer convention $\mathrm{nint}(x) = \lfloor x + 1/2 \rfloor$ in this analysis. The integer lifts are fixed before taking residues. If implementation rounding differs, the discrepancy must be covered by an explicit sampler/arithmetic certification rather than ignored.

Lemma 34 (Normalized stochastic quantizer law and inverse). *Suppose Q is odd, $K = 2^c$, and X, ω are independent uniform variables in $\mathbb{Z}_Q$. Write $KX = Qb + r$ with $0 \le r < Q$ and $C = b + \beta \pmod K$, where $\Pr[\beta = 1 \mid r] = r/Q$. Set $E = Q\beta - r$. Then*

$$\Pr[E = e] = \frac{Q - |e|}{Q^2}, \qquad |e| \le Q - 1, \tag{63}$$

which is the distribution of $A - B$ for independent $A, B \leftarrow U(\{0, \dots, Q-1\})$. The reconstruction obeys

$$\mathsf{DRec}_K(C) - X = \mathrm{nint}(E/K) \pmod Q.$$

Given E, a uniform PPT procedure reconstructs (X, C, ω) with exactly its original conditional law.

Proof. For $e = -r \le 0$ the event $\beta = 0$ has probability $(Q - r)/Q^2$; for $e = Q - r > 0$ the event $\beta = 1$ has probability r/Q^2. Both give (63). For nonpositive e, set $(\beta, r) = (0, -e)$; for positive e, set $(\beta, r) = (1, Q - e)$. Since Q is odd and K is a power of two, $K^{-1} \pmod Q$ exists, and $X = K^{-1}r \pmod Q$. Recompute $b = (KX - r)/Q$ and $C = b + \beta \pmod K$. Finally choose ω uniformly from the subinterval of $\mathbb{Z}_Q$ giving the stipulated rounding event. This is nonempty on the support of E and reproduces the exact conditional law. The reconstruction identity follows by substituting $KC = KX - r + Q\beta$ before nearest-integer decoding.

The corrupted client's response-quantization coin is included in $\mathsf{CorSt}_{\mathcal{A}}$ rather than sampled anew. This requires a separate inverse distribution.

Lemma 35 (Fixed-coin response primitive). *Fix the response-quantizer coin $\omega_j^{(2)} \in \{0, \dots, Q-1\}$ of a corrupted client j. In the deferred uniform-DD reference game, the corresponding c_2 primitive $E_j^{(2)} = Q\beta - r$ is uniform on $\{-\omega_j^{(2)}, \dots, Q - 1 - \omega_j^{(2)}\}$. Writing*

$$\mu_j = (Q-1)/2 - \omega_j^{(2)}, \qquad \bar{E}_j^{(2)} = E_j^{(2)} - \mu_j,$$

we obtain a centered uniform integer on $\{-(Q-1)/2, \dots, (Q-1)/2\}$. Given this centered integer and the fixed coin, its X_j and $C_j^{(2)}$ are uniquely determined.

Proof. For fixed ω, the quantizer in Section 4.4 chooses $\beta = 0$ precisely when $r \le \omega$; hence $r \mapsto Q\beta - r$ permutes $\{-\omega, \dots, Q - 1 - \omega\}$. Uniform DD makes the input X_j uniform and therefore $r = K_2 X_j \bmod Q$ uniform. The inverse is $(\beta, r) = (0, -E)$ for $E \le 0$ and $(\beta, r) = (1, Q - E)$ for $E > 0$, followed by $X_j = K_2^{-1} r \bmod Q$ and the encoded code $C_j^{(2)} = (K_2 X_j - r)/Q + \beta \pmod{K_2}$.

Centered score and exact finite error. For one released coefficient put $I_H = I_H(\nu)$, $a = |\mathcal{A}|$ and $h = |\mathcal{H}|$. Honest c_1 triangular primitives are indexed by $i \in I_H$, honest c_2 triangular primitives by $j \in \mathcal{H}$, and centered corrupted c_2 uniform primitives by $j \in \mathcal{A}$. Denote their tuple by $\xi = (\xi_1, \xi_2)$ and its product distribution by $p(\xi)$. Set

$$\mu_{\mathcal{A}} = \sum_{j \in \mathcal{A}} \mu_j, \qquad t_{\mathcal{A}} = \operatorname{nint}(\mu_{\mathcal{A}}/K_2).$$

The exact (finite) integer error after centering is

$$\begin{aligned} U_1(\xi_1) &= \sum_{i \in I_H} \alpha_i \operatorname{nint}(E_i^{(1)}/K_1), \\ U_2(\xi_2) &= \operatorname{nint}\left(\frac{\sum_{j \in \mathcal{H}} E_j^{(2)} + \sum_{j \in \mathcal{A}} \bar{E}_j^{(2)} + \mu_{\mathcal{A}}}{K_2} \right) - t_{\mathcal{A}}, \\ U_{\mathrm{fin}}(\xi) &= U_1(\xi_1) + U_2(\xi_2). \end{aligned} \tag{64}$$

Here $t_{\mathcal{A}}$ is absorbed into the known offset $k_{\mathcal{A}}^{\mathrm{out}}$ in (62). In particular, U_2 implements *one reconstruction of the sum of the response codes*, not a sum of individually reconstructed c_2 errors. Define the linear score

$$L(\xi) = \sum_{i \in I_H} \alpha_i E_i^{(1)}/K_1 + \frac{\sum_{j \in \mathcal{H}} E_j^{(2)} + \sum_{j \in \mathcal{A}} \bar{E}_j^{(2)}}{K_2}.$$

Lemma 36 (Uniform rounding gap). *For every primitive tuple ξ,*

$$|U_{\mathrm{fin}}(\xi) - L(\xi)| \le D_{\mathrm{rnd}} := \tfrac{1}{2} \sum_{i \in I_H} \alpha_i + \tfrac{1}{2} + \tfrac{1}{2} \mathbf{1}_{\{a>0\}} \le 65.$$

Proof. Each nearest-integer operation changes its real input by at most $1/2$. The honest c_1 term contributes $\frac{1}{2} \sum_{i \in I_H} \alpha_i$, the single c_2 code-sum reconstruction contributes $1/2$, and the centering correction $t_{\mathcal{A}}$ contributes at most $1/2$ when $a > 0$. The setup-fixed positive weights sum to 128 over all clients.

The prescribed centered residual. The decryption identity is initially a congruence modulo odd Q. For each coefficient define, without assuming no wrap,

$$R = \mathsf{Center}_Q(O - M - k_{\mathcal{A}}^{\mathrm{out}}) \in [-(Q-1)/2, (Q-1)/2] \cap \mathbb{Z}. \tag{65}$$

Then $R = \mathsf{Center}_Q(U_{\mathrm{fin}} + G_\Sigma)$ always, where $G_\Sigma = \sum_{j \in \mathcal{H}} G_j$. The integer equation $R = U_{\mathrm{fin}} + G_\Sigma$ holds only on the good centered-lift branch; its exceptional probability is charged *once* below.

7.3 Exponential Tilting and Exact Conditional Sampling

For a fixed coefficient, let $G_j \leftarrow D_{\mathbb{Z},\sigma}$ be independent honest smudging variables. Write $s^2 = h\sigma^2$ and let G_s be a reference $D_{\mathbb{Z},s}$ sample. We first sample the conditional law with G_s replacing G_Σ; the cost of this replacement and of splitting

G_s into honest shares is tracked in Subsection 7.5. For R on the good branch, the target distribution of primitive tuple ξ and integer g is

$$\pi_R(\xi, g) \;\propto\; p(\xi) \exp\left(-\frac{g^2}{2s^2}\right) \mathbf{1}\{U_{\text{fin}}(\xi) + g = R\}. \tag{66}$$

What the conditional sampler must reproduce. For an output residue with good centered lift, the unknown finite primitive tuple is $\xi = (\xi_1, \xi_2)$ from (64). Its unconditioned law is $p(\xi)$. In the width-s *reference* model (66), its conditional law for the supplied residual R is

$$\Pr_{\text{ref}}[\xi = x \mid R] = \frac{p(x) \exp[-(R - U_{\text{fin}}(x))^2/(2s^2)]}{\sum_{x'} p(x') \exp[-(R - U_{\text{fin}}(x'))^2/(2s^2)]}, \qquad s^2 = h\sigma^2. \tag{67}$$

Here the denominator ranges over the finite support of the quantization primitives. This is the concrete target of TILT: R determines the Gaussian-sum candidate $g = R - U_{\text{fin}}(x)$, and its probability is weighted by the Gaussian mass at that g. Sampling x from $p(x)$ and merely setting g to this difference would omit the exponential factor. The algorithm below samples this law without enumerating its exponentially large finite support. The replacement of the true Gaussian sum by the width-s reference law is *not* free: its cost is recorded in (78).

It would not be sound to draw $\xi \sim p$ and then set $g = R - U_{\text{fin}}(\xi)$ without accounting for this Gaussian weight; doing so would change the conditioned transcript law.

A rational tilt proposal. For any finite $\theta \in \mathbb{R}$, define

$$\begin{aligned} &Z_L(\theta) = \mathbb{E}_p e^{\theta L(\xi)}, && q_\theta(\xi) = \frac{p(\xi) e^{\theta L(\xi)}}{Z_L(\theta)}, \\ &\mu_L(\theta) = \mathbb{E}_{q_\theta} L, && V_L(\theta) = \operatorname{Var}_{q_\theta} L, \\ &F_R(\theta) = s^2\theta + \mu_L(\theta) - R. \end{aligned} \tag{68}$$

Because $F_R'(\theta) = s^2 + V_L(\theta) > 0$ and $s^2\theta$ dominates at the tails, F_R has a unique root. *This root need not be represented exactly.* The algorithm uses any finite rational $\widetilde{\theta}$ satisfying the explicit acceptance and running-time certificates below. The exactness of the target posterior does not depend on how closely $\widetilde{\theta}$ approximates the root.

Lemma 37 (Factorization of the tilted proposal). *For every finite rational $\widetilde{\theta}$, a sample from $q_{\widetilde{\theta}}$ can be generated from at most $2|I_H| + 2h + a \le 4h + a$ independent truncated geometric variables on $\{0, \dots, Q-1\}$, each with ratio $\exp(\widetilde{\theta} w)$ for a publicly known signed weight w. With certified finite-precision CDF and inverse-CDF routines the resulting proposal can be sampled in time polynomial in the bit lengths and the required precision.*

Proof. Each honest triangular $E = A - B$ is a difference of two independent discrete uniform integers. Multiplying its probability by $e^{\widetilde{\theta} w (A-B)}$ tilts A and B

into independent truncated geometric variables with opposite ratios. A centered corrupted response primitive is one uniform variable with a known translation, so contributes one geometric factor. Normalization constants are finite geometric sums; with outward interval rounding and binary search they admit inverse-CDF sampling without enumerating Q states. The claimed bit complexity requires the separately stipulated arithmetic-precision certificate.

Put $d(\xi) = U_{\mathrm{fin}}(\xi) - L(\xi)$, set $g = R - U_{\mathrm{fin}}(\xi)$, and define

$$a_{\widetilde{\theta}}(\xi) = \exp\left(-\frac{(g - s^2\widetilde{\theta})^2}{2s^2} + \widetilde{\theta}d(\xi) - |\widetilde{\theta}|D_{\mathrm{rnd}}\right). \tag{69}$$

Lemma 38 (Exact accepted distribution for any rational tilt). *For every finite rational $\widetilde{\theta}$, $0 < a_{\widetilde{\theta}}(\xi) \le 1$. Sampling $\xi \sim q_{\widetilde{\theta}}$ and accepting with probability $a_{\widetilde{\theta}}(\xi)$ produces exactly the posterior* (66) *in ideal exact arithmetic. Thus the error of approximating the saddlepoint by a rational number is* zero*; the effects of approximate proposal and Bernoulli implementation are separately accounted for.*

Proof. The squared term in the exponent of (69) is nonpositive, and $\widetilde{\theta}d - |\widetilde{\theta}|D_{\mathrm{rnd}} \le 0$ by Lemma 36. Moreover,

$$\begin{aligned}&\widetilde{\theta}L - \frac{(g - s^2\widetilde{\theta})^2}{2s^2} + \widetilde{\theta}d - |\widetilde{\theta}|D_{\mathrm{rnd}}\\ &\quad= -\frac{g^2}{2s^2} + \widetilde{\theta}R - \frac{s^2\widetilde{\theta}^2}{2} - |\widetilde{\theta}|D_{\mathrm{rnd}}.\end{aligned}$$

The final three terms depend on the prescribed R and $\widetilde{\theta}$, but not on ξ; multiplication by $q_{\widetilde{\theta}}(\xi)$ and normalization therefore gives (66).

Direct check of the accepted weight. Combining (68) and (69) gives the identity

$$q_{\widetilde{\theta}}(x)a_{\widetilde{\theta}}(x) = C(R,\widetilde{\theta})p(x)\exp\left(-\frac{(R - U_{\mathrm{fin}}(x))^2}{2s^2}\right), \tag{70}$$

where the factor $C(R,\widetilde{\theta}) > 0$ does not depend on x. Thus the rejection step reproduces (67) after normalization. This algebraic identity proves correctness of the *ideal* accepted law; the subsequent acceptance-rate and finite-precision certificates, not this identity, establish an efficient implemented sampler.

Lemma 39 (Certified one-trial acceptance probability). *For any finite $\widetilde{\theta}$, set*

$$\Gamma_R(\widetilde{\theta}) = \frac{\left(|F_R(\widetilde{\theta})| + \sqrt{V_L(\widetilde{\theta})} + D_{\mathrm{rnd}}\right)^2}{2s^2} + 2D_{\mathrm{rnd}}|\widetilde{\theta}|.$$

Then the ideal one-trial acceptance probability satisfies $A_R \ge \exp(-\Gamma_R)$.

Proof. Writing $\mu = \mu_L(\widetilde{\theta})$, $g - s^2\widetilde{\theta} = -F_R(\widetilde{\theta}) - (L - \mu) - d$. Jensen's inequality for the logarithm of the acceptance expectation and the L_2 Minkowski inequality give

$$\begin{aligned}\log A_R &\geq -\frac{\mathbb{E}_{q_{\widetilde{\theta}}}(g - s^2\widetilde{\theta})^2}{2s^2} - 2D_{\mathrm{rnd}}|\widetilde{\theta}| \\ &\geq -\Gamma_R(\widetilde{\theta}).\end{aligned}$$

This completes the bound.

7.4 Uniform Polynomial-Time Acceptance Certificates

A rejection sampler with an uncontrolled one-trial acceptance probability need not be polynomial time. We therefore require a *uniform* certificate over all output residues and contributor sets that can occur in the declared execution. Write η_{root} for the certified residual of a finite rational tilt: $|F_R(\widetilde{\theta}_R)| \leq \eta_{\mathrm{root}}$.

Lemma 40 (Symmetry and a root-independent tilt bound). *The centered linear score L has a symmetric distribution. Hence $\mu_L(0) = 0$, μ_L is odd and nondecreasing, and for every certified rational tilt,*

$$|\widetilde{\theta}_R| \leq \frac{|R| + \eta_{\mathrm{root}}}{s^2}.$$

Proof. Every difference of two uniform variables and every centered corrupted primitive is symmetric. Their independent weighted sum has the same law as its negation. Consequently $Z_L(\theta)$ is even and $\mu_L = (\log Z_L)'$ is odd with nonnegative derivative V_L. Since $\widetilde{\theta}_R\mu_L(\widetilde{\theta}_R) \geq 0$, combining $s^2\widetilde{\theta}_R + \mu_L(\widetilde{\theta}_R) = R + F_R(\widetilde{\theta}_R)$ with the sign of $\widetilde{\theta}_R$ yields the claimed inequality.

Uniform residual cutoff. For the current released coefficient let $h = |\mathcal{H}|$, $a = |\mathcal{A}|$, and let I_H be its honest upload contributors as defined after Lemma 32. Using the centered independent primitive representation, define

$$C_L(I_H) = (Q-1)^2\left(\frac{\sum_{i\in I_H}\alpha_i^2}{4K_1^2} + \frac{h}{4K_2^2} + \frac{a}{8K_2^2}\right).$$

Hoeffding's lemma gives $\log \mathbb{E}e^{uL} \leq C_L u^2$. Since $|U_{\mathrm{fin}} - L| \leq D_{\mathrm{rnd}}$, combining the Gaussian proxy and the score bound yields, for $B > D_{\mathrm{rnd}}$,

$$\Pr[|U_{\mathrm{fin}} + G_s| > B] \leq 2\exp\left(-\frac{(B - D_{\mathrm{rnd}})^2}{2s^2 + 4C_L}\right). \tag{71}$$

For a chosen per-coordinate output-tail allocation $\delta_{\mathrm{out}}^{(1)}$, define the public cutoff

$$B_{\mathrm{cond}} = \left\lceil D_{\mathrm{rnd}} + \sqrt{(2s^2 + 4C_L)\log\frac{2}{\delta_{\mathrm{out}}^{(1)}}}\right\rceil < Q/2. \tag{72}$$

The inequalities must be checked for the model and auxiliary parameters separately. The same monotonicity permits an all-honest $I_H = \mathcal{H}$ envelope for Sparse. A certified Fourier/Poisson acceptance estimate may replace the Jensen bound, but only with a separately proved *contributor-set-uniform* lower bound or a polynomial-size interval certificate; evaluating merely $I_H = \mathcal{H}$ is not sufficient for that route.

Lemma 41 (Uniform variance and Jensen envelope). *Let $h = |\mathcal{H}|$, $a = |\mathcal{A}|$ and $I_H \subseteq \mathcal{H}$ in Sparse; on Full and auxiliary paths $I_H = \mathcal{H}$. Define*

$$V_*(I_H) = \frac{(Q-1)^2}{4}\left(\frac{2\sum_{i\in I_H}\alpha_i^2}{K_1^2} + \frac{2h+a}{K_2^2}\right). \tag{73}$$

For every finite tilt θ, $V_L(\theta) \le V_*(I_H)$. For all $|R| \le B_{\mathrm{cond}}$ with certified $|F_R(\widetilde{\theta}_R)| \le \eta_{\mathrm{root}}$,

$$\Gamma_R \le \Gamma_{\max}(I_H) := \frac{\left(\eta_{\mathrm{root}} + \sqrt{V_*(I_H)} + D_{\mathrm{rnd}}\right)^2}{2s^2} + \frac{2D_{\mathrm{rnd}}(B_{\mathrm{cond}} + \eta_{\mathrm{root}})}{s^2}. \tag{74}$$

All terms are nondecreasing when I_H is enlarged to $\mathcal{H}$ with nonnegative weights, when B_{cond} is computed by the preceding cutoff (72)*. Thus $I_H = \mathcal{H}$ gives a* uniform sufficient certificate *for every admissible Sparse contributor subset; no exponential enumeration is required for the Jensen route.*

Proof. The tilted triangular law factors into two truncated geometric variables on intervals of length $Q-1$, and the centered corrupted primitive contributes one. Popoviciu's inequality bounds the variance of any variable supported on an interval of length D by $D^2/4$, independently of its tilt. Variances add under the independent tilted factors and give (73). Lemma 40 bounds $|\widetilde{\theta}_R|$ by $(B_{\mathrm{cond}} + \eta_{\mathrm{root}})/s^2$. Substitute these inequalities into Lemma 39. Positive weights make the sums and their squares monotone under enlargement of the contributor set. This argument does *not* claim that an alternative Fourier acceptance bound is monotone in I_H.

Bounding rejection and arithmetic error. For all certified residues with $|R| \le B_{\mathrm{cond}}$, choose $\Gamma_{\max}$ from (74) or from an independently certified uniform alternative and set

$$T_{\max} = \left\lceil e^{\Gamma_{\max}} \log \frac{1}{\delta_{\mathrm{rej}}^{(1)}} \right\rceil. \tag{75}$$

The required *hard* condition is $T_{\max} \le T_{\mathrm{policy}} = \mathrm{poly}(\lambda)$, not merely a finite expected running time. The finite-precision implementation ledger, including unresolved comparison failures, is specified immediately below. The root residual, geometric-CDF intervals, acceptance Bernoulli and exponentiation are all to be certified by outward-rounded finite-precision arithmetic. A rational root approximation incurs no separate TV term; a failure of any of these tests makes the candidate *uncertified*, not silently accepted.

Finite-precision TILT implementation interface. A concrete implementation of the ideal sampler in Lemma 38 is specified by the following bounded operations. Unlike the ideal posterior identity, their running-time guarantee requires the stated *certificates*.

1. **Rational tilt.** Using bracketing and bisection on F_R, compute a rational $\widetilde{\theta}$ and certified intervals for μ_L, V_L such that $|F_R(\widetilde{\theta})| \le \eta_{\mathrm{root}}$. Monotonicity follows from $F_R' = s^2 + V_L > 0$. The permitted bit precision and the maximum bisection steps are prescribed before execution.
2. **Geometric proposal.** For each factor on $\{0, \dots, Q-1\}$, use the finite geometric partition function $Z(r) = \sum_{j=0}^{Q-1} r^j$, $r = e^{\widetilde{\theta} w}$, and inverse-CDF binary search. Compute comparisons in the log or rescaled domain using interval enclosures; the routine must provide a uniform TV error η_{geom} using $\mathrm{poly}(\log Q, b)$ bit operations for a certified precision budget b. Ambiguous interval comparisons at the precision cap are reported as failures, not resolved by guessing.
3. **Bernoulli accept/reject.** Evaluate the negative exponent in (69) by certified rational intervals and compare it to a finite random binary fraction. The exponential-Bernoulli procedure must have uniform error at most η_{exp} and a polynomial bit-operation cap.
4. **Bounded outer loop.** Repeat at most $T_{\max}$ trials. An unresolved comparison or an exhausted trial cap enters the explicitly defined exceptional constructor below.

For $N_f \le 4h + a$ geometric factors per trial, coupling every finite operation to its ideal counterpart gives the conditional *unconditioned-in-trials* implementation ledger

$$\delta^{(1)}_{\mathrm{arith}} \le T_{\max}(N_f \eta_{\mathrm{geom}} + \eta_{\mathrm{exp}}) + \delta^{(1)}_{\mathrm{precision}},$$

where $\delta^{(1)}_{\mathrm{precision}}$ covers the certified probability of ambiguous comparisons at the cap, if not already absorbed into the two per-call bounds. One must use *one* accounting convention, without double-charging. The expression is a sufficient certificate interface; it does not assert that unspecified real-number operations are executable in uniform polynomial time. A concrete candidate must supply polynomial precision, bisection, geometric inversion, Bernoulli, and trial bounds. Table 7 reports the candidate-specific static assessment for Full and Sparse $k = 4$. This evidence does not certify arbitrary parameter choices.

7.5 Gaussian Convolution, Centered Lifts, and Share Splitting

The reference density in (66) uses a single discrete Gaussian of width $s = \sqrt{h}\sigma$. In the actual protocol the smudging value is the sum of h independent $D_{\mathbb{Z},\sigma}$ samples. It is important not to assert that these laws are identical: convolution of discrete Gaussians differs slightly from the nominal Gaussian on $\mathbb{Z}$. We include both this difference and the approximate conditional share-splitting cost in the error ledger.

Lemma 42 (Discrete-Gaussian convolution certificate). *Let $h \geq 2$, $\sigma \geq \frac{1}{2}$, and let the h integer-valued smudging variables be independent and identically distributed as $D_{\mathbb{Z},\sigma}$, with mass proportional to $\exp(-z^2/(2\sigma^2))$. For $G_\Sigma = \sum_{i=1}^{h} G_i$ and an independent reference $G_s \sim D_{\mathbb{Z},\sqrt{h}\sigma}$, set*

$$S_h(\sigma) := \sum_{j=1}^{h-1} \exp\left(-2\pi^2\sigma^2 \frac{j}{j+1}\right). \tag{76}$$

The Kairouz–Liu–Steinke discrete-Gaussian convolution result (ICML 2021, supplementary Theorem 11 and Corollary 12) [6] provides the bidirectional *pointwise ratio bound*

$$\sup_{z\in\mathbb{Z}} \left|\log \frac{\Pr[G_\Sigma = z]}{\Pr[G_s = z]}\right| \leq 5S_h(\sigma). \tag{77}$$

In particular the deliberately looser certificate used here is

$$\Delta_{\mathrm{TV}}(G_\Sigma, G_s) \leq \delta^{(1)}_{\mathrm{conv}} := \min\{1, e^{2\tau_{\mathrm{conv}}} - 1\}, \qquad \tau_{\mathrm{conv}} = 10S_h(\sigma). \tag{78}$$

For $h = 1$ the two laws coincide; by convention the sum in (76) *is empty and the cost is zero.*

Proof. For a two-term convolution with widths $\sigma_1, \sigma_2 \geq 1/2$, the cited Theorem 11 bounds the pointwise logarithmic ratio to the nominal Gaussian of width $\sqrt{\sigma_1^2 + \sigma_2^2}$ by $5\exp[-2\pi^2/(\sigma_1^{-2} + \sigma_2^{-2})]$. Its Corollary 12 applies this result inductively with $\sigma_1 = \sqrt{j}\sigma$, $\sigma_2 = \sigma$; post-processing preserves the divergence bound and summing $j = 1, \ldots, h-1$ gives (77). Both the cited result and our reference distribution use $\exp(-z^2/(2\sigma^2))$, with σ a standard-deviation parameter, rather than a lattice-Gaussian $s = \sqrt{2\pi}\sigma$ parameter. If $|\log(P(z)/Q(z))| \leq u$ pointwise, then $\sum_z |P(z) - Q(z)|/2 \leq (e^u - 1)/2$, and hence also $\Delta_{\mathrm{TV}}(P, Q) \leq \min\{1, e^{2u} - 1\}$. Our chosen $\tau_{\mathrm{conv}} = 10S_h \geq 5S_h$ is thus a valid conservative replacement. This comparison is an *unconditional sum-law* comparison, not a statement that conditioning on a particular rare value of G_Σ preserves the same distance. The subsequent output-conditioning operation must be carried out on the joint law as in Lemma 48.

Lemma 43 (One-time centered-lift replacement). *Let $Z = U_{\mathrm{fin}} + G_\Sigma \in \mathbb{Z}$ and $R = \mathsf{Center}_Q(Z)$. Suppose $B_{\mathrm{cond}} < Q/2$ and $\Pr[|Z| > B_{\mathrm{cond}}] \leq \delta^{(1)}_{\mathrm{out}}$. Replacing the residue-conditioned suffix on the good branch by the integer-conditioned suffix $Z = R$ changes the unconditional joint law by at most $\delta^{(1)}_{\mathrm{out}}$. An exceptional residue must be handled by an explicit output-preserving fallback; there is no additional independent "wrap event" to count.*

Proof. For each centered residue r, split the mass assigned to it into $\Pr[Z = r]$ and the alias mass $\sum_{\ell\neq 0} \Pr[Z = r + \ell Q]$. All aliases and all out-of-cutoff residues together have total probability bounded by $\Pr[|Z| > B_{\mathrm{cond}}]$. Coupling identical samples on the complement and replacing only this exceptional mass gives the claimed TV bound. In particular the comparison is not obtained by conditioning the *entire* execution on a global no-wrap event.

For a prescribed integer sum g, the exact conditional joint law of the honest Gaussian shares is

$$\Pr[(G_1,\dots,G_h)=\mathbf{g} \mid \textstyle\sum_i G_i = g] \;\propto\; \exp\!\Big(-\tfrac{\sum_i g_i^2}{2\sigma^2}\Big)\,\mathbf{1}\{\textstyle\sum_i g_i = g\}. \tag{79}$$

It is a discrete Gaussian on the affine coset $\mathbf{z}_0 + A_{h-1}$, where $A_{h-1} = \{\mathbf{z} \in \mathbb{Z}^h : \sum_i z_i = 0\}$.

Lemma 44 (Certified affine-A_{h-1} conditional splitter). *Let $r = h-1$, $q_\sigma = e^{-\pi^2\sigma^2}$ and $\beta_\sigma = 2q_\sigma/(1-q_\sigma) < 1$. Assume a one-dimensional shifted-integer Gaussian sampler with uniform TV error at most η_{1D} for all centers and widths needed below. The backwards Klein/GPV sampler on the public basis $\mathbf{b}_j = \mathbf{e}_j - \mathbf{e}_h$ generates an approximation to* (79) *with*

$$\delta^{(1)}_{\mathrm{split}} \le \min\left\{1, \frac{e^{\tau_{\mathrm{coset}}}-1}{2}\right\} + (h-1)\eta_{1D}, \quad \tau_{\mathrm{coset}} = (h-1)\log\frac{1+\beta_\sigma}{1-\beta_\sigma}. \tag{80}$$

This bound is uniform over every integer aggregate sum g.

Proof. The Gram–Schmidt squares of $(\mathbf{e}_1-\mathbf{e}_h,\dots,\mathbf{e}_{h-1}-\mathbf{e}_h)$ are $(j+1)/j \le 2$. With the $e^{-\|z\|^2/(2\sigma^2)}$ convention, the GPV width is $\sqrt{2\pi}\sigma$, so the sequential one-dimensional widths are at least $\sqrt{\pi}\sigma$. For a shifted one-dimensional lattice Gaussian, Poisson summation bounds its normalized theta sum relative to its leading continuous factor by $1 \pm \beta_\sigma$, uniformly over the center. Multiplying the r sequential normalization ratios bounds the pointwise likelihood ratio between the exact coset Gaussian and the idealized backwards sampler by $\exp(\pm r\log((1+\beta_\sigma)/(1-\beta_\sigma)))$. The same max-divergence to TV argument as above gives the first term of (80). Coupling the at most r approximate one-dimensional sampling calls adds at most $r\eta_{1D}$. All estimates depend on widths and dimension, not on g.

Per-coefficient certificate. Combining the three departures from the ideal integer-posterior sampler with its rejection and arithmetic errors yields

$$\delta^{(1)}_{\mathrm{cond}} := \delta^{(1)}_{\mathrm{conv}} + \delta^{(1)}_{\mathrm{out}} + \delta^{(1)}_{\mathrm{rej}} + \delta^{(1)}_{\mathrm{split}} + \delta^{(1)}_{\mathrm{arith}}. \tag{81}$$

The Gaussian parameters, K_1, K_2 and error budgets are path-specific. The formula deliberately charges the no-wrap replacement only once. The original smudging and the linearized finite-error primitives are not asserted to be identically distributed; rather, all replacements are accompanied by their explicit TV bounds.

The coefficient-level conditional kernel in order. For each released coefficient, the objects passed between the finite-code, TILT and Gaussian subroutines are concretely

$$(M, O, k^{\mathrm{out}}_{\mathcal{A}}) \xrightarrow{(65)} R \xrightarrow{(66)} (\xi, g) \xrightarrow{(79)} (\xi, G_1, \dots, G_h). \tag{82}$$

The middle arrow means the certified TILT rejection sampler, not an oracle that samples from an arbitrary conditioned law. The last arrow conditions the h honest smudging shares on $\sum_{i=1}^{h} G_i = g$; it never changes the contributor set I_H of the *upload* error in (64). Next, the inverse kernels of Lemmas 34 and 35 turn ξ into the actual quantization code/coin tuples. Only after these values exist are the cached-key server computations replayed. Finally Lemma 46 turns those raw responses into DD-compatible code inputs. This ordering is essential: choosing the response codes independently would generally violate the prescribed output equation (18).

7.6 Constructing the Entire Output-Conditioned Suffix

The object constructed here is *not* only the scalar tuple (ξ, g) of (82): it is the complete server–colluder suffix, including all transmitted codes, server intermediates and partial-decryption responses. For one released instance ν, write

$$\mathsf{BuildSuffix}_\nu(\mathsf{SetupSt}, \mathsf{Hist}_{<\nu}, A_\nu, \mathsf{CorSt}_{\mathcal{A}}, M_\nu, O_\nu) \longmapsto \mathsf{Suffix}_\nu.$$

The setup and the input layout A_ν have the meanings of Subsection 7.1. Write $\mathsf{CosetSplit}(g)$ for the certified conditional-share sampler of Lemma 44; its output is an h-tuple of integers summing to g. The object Suffix_ν comprises the actual c_1/c_2 codes, aggregate b, responses, public DD values and raw output. The construction is applied coefficientwise to *released* positions, but the same packed ciphertext, mask, cached row and seed are shared wherever required by the protocol. In particular, R2M must be applied to a packed RLWE ciphertext before its k lanes can be processed separately.

1. **Generate the input-independent prefix.** Draw $\mathsf{SetupSt}$ *once* for the execution; sample the public base-mask seeds and identifiers B_ν from their specified setup/round laws, then draw $\mathsf{Frontier}_\nu \leftarrow \mu_\nu(\cdot \mid \mathsf{SetupSt}, A_\nu, \mathsf{Hist}_{<\nu}, B_\nu)$ by Lemma 31. This fixes the aggregate-a input a_ν, its compressed code and $\widehat{a}_\nu$. No c_1 suffix, aggregate b or response code is included in this prefix.
2. **Solve the output constraint.** For every released coefficient compute

$$\begin{aligned} R_\nu &= \mathsf{Center}_Q(O_\nu - M_\nu - k^{\mathrm{out}}_{\mathcal{A},\nu}), \\ (\xi_\nu, g_\nu) \leftarrow \pi_{R_\nu}, &\qquad g_\nu = R_\nu - U_{\mathrm{fin}}(\xi_\nu), \\ (G_{i,\nu})_{i\in\mathcal{H}} \leftarrow \mathsf{CosetSplit}(g_\nu), &\quad \sum_{i\in\mathcal{H}} G_{i,\nu} = g_\nu. \end{aligned} \tag{83}$$

   The first line is (65); the second uses the certified TILT law (66) and the exact finite error (64); the last line invokes Lemma 44. On the certified exceptional branch replace this sampling step by Lemma 45, not by an unconditioned choice of ξ_ν.
3. **Recover the actual c_1 upload codes.** Apply the inverse of Lemma 34 to the c_1 primitives within ξ_ν; this produces the quantizer input, valid code and associated coin for each *transmitted* honest coefficient. Restore the appropriate $P_{S_{i,j}} b_{i,j}$ positions, zero-fill absent Sparse lanes, and reconstruct the

aggregate-upload input $\widehat{b}$. This step must use the *same* support $S_{i,j}$ fixed in A_ν; no support is redrawn after observing O_ν.

4. **Recompute the server's b-side arithmetic.** Apply, in protocol order, KSwitch, R2M (when $k > 1$), label-wise weighted aggregation and M2R (when $k > 1$), using the cached keys and the frontier's already fixed stochastic digits/ModDown coins. This determines the aggregate b_ν. The a-side $a_\nu, \widehat{a}_\nu$ is *not* resampled; it was determined in the first step.
5. **Produce all partial-decryption codes.** For each nonpivot honest i let $z_{i,\nu}^{\mathrm{sim}}$ denote its simulated share of the decryption key: t_i^{sim} for Sparse model $k > 1$, and s_i^{sim} for Full, Sparse $k = 1$, or auxiliary outputs. Form its raw response using the individual identity of Subsection 5.4:

$$h_{i,\nu} = \widehat{a}_\nu z_{i,\nu}^{\mathrm{sim}} + G_{i,\nu}.$$

   Use Lemma 24 for the pivot, without requiring the challenge secret. For corrupted i, retain the given $\mathsf{CorSt}_{\mathcal{A}}$ and use the fixed-coin inverse of Lemma 35; for honest i, use Lemma 34. The generated codes $C_{i,\nu}^{(2)}$ cover *all* $i \in \mathcal{C}$, including noncontributors to a Sparse upload. No response code is selected independently of the finite tuple ξ_ν.
6. **Restore DD and check the raw output.** For the selected prequantization inputs $X_{i,\nu}$, use Lemma 46 to set

$$D_{i,\nu} = X_{i,\nu} - h_{i,\nu} - \mathbf{1}_{\{i=o^\star\}}\mathsf{Off}_\nu \pmod{Q}.$$

   Sum the c_2 codes *before* reconstructing once, and verify the exact wire-level identity

$$b_\nu + \mathsf{DRec}_{K_2}\left(\sum_{i\in\mathcal{C}} C_{i,\nu}^{(2)} \bmod K_2\right) - \mathsf{Off}_\nu - \sum_{i\in\mathcal{C}} D_{i,\nu} \equiv O_\nu \pmod{Q}. \quad (84)$$

   The deferred polynomial dither is lifted to a canonical ideal-XOF byte stream *after* the suffix is fixed, under Lemmas 46 and 47.

The normal branch in this list uses an approximately sampled conditional distribution, with the per-coefficient error (81); it is not a claim of exact sampling from the true unbounded joint Gaussian sum. The exceptional branch instead uses the separate algebraic constructor below. In *both* branches the recovered output is exactly O_ν; only the former is close to the terminal conditional transcript distribution in the certified joint-law sense.

Lemma 45 (Explicit output-preserving exceptional constructor). *Fix a valid setup/frontier and permitted semantic layout, all corrupted-client long-term states and private quantizer coins, and an arbitrary prescribed raw released coefficient $o \in \mathbb{Z}_Q$. In the* flooded terminal *polynomial-reference game there is a deterministic polynomial-time construction of a syntactically valid suffix satisfying* $\mathsf{Recover} = o$ *exactly, without replacing a corrupted client's private state. Its output is* not *claimed to have the genuine conditional distribution. The construction is used only on an exceptional branch, whose* unconditional joint-law mass *is charged to* $\delta_{\mathrm{out}}^{(1)} + \delta_{\mathrm{rej}}^{(1)} + \delta_{\mathrm{arith}}^{(1)}$.

Proof. Choose a fixed valid c_1 terminal upload-code/coin pair for each honest transmitted coefficient, zero-fill absent Sparse lanes, and replay all cached-key switching, R2M and M2R operations with *the already fixed frontier coins*. This determines the actual aggregate b and its transported c_1 error U_1. Fix the corrupted native raw partial responses from their supplied long-term state and private smudging coins; choose fixed valid smudging integers for every nonpivot honest user, e.g., zero. Write $G_{\neg p^\star}$ for their sum, including corrupted smudging integers. Select a signed integer $g_{p^\star}$ representing

$$g_{p^\star} \equiv o - M - U_1 - G_{\neg p^\star} \pmod{Q}.$$

Because the integer discrete Gaussian has full support on $\mathbb{Z}$, this is a legal *support point*, although it need not be likely. Generate the flooded pivot raw response by (57). The phase identity then gives $b + \sum_{i\in\mathcal{C}} h_i \equiv o \pmod{Q}$.

Now choose the deferred DD offset $\mathsf{Off} = 0$ and the pre-quantization response input $X_i = 0$ for *every* client. Set $D_i = -h_i \pmod{Q}$; these public dithers satisfy $X_i = h_i + D_i + \mathbf{1}_{\{i=o^\star\}}\mathsf{Off}$. The normalized quantizer satisfies $\mathsf{SQ}_{K,\omega}(0) = 0$ for *every* $\omega \in \mathbb{Z}_Q$, including each corrupted client's *fixed* private coin. Thus all valid c_2 codes are zero. Since $\mathsf{DRec}_{K_2}(0) = 0$, the prescribed code-sum reconstruction yields

$$b + \mathsf{DRec}_{K_2}(0) - \mathsf{Off} - \sum_i D_i = b + \sum_i h_i = o \pmod{Q}.$$

This operation is coefficientwise over the actually released positions and respects the unique packed layout; the remaining unreleased positions can be assigned valid canonical values.

The DD values are initially chosen in the delayed-polynomial reference game. For byte-level lifting, choose an as-yet-unqueried DD root and program each domain-separated residue address with a valid accepted rejection word representing the target CRT residue. A root absent from the at most q_X^{DD} prequeried roots can be located by examining at most $q_X^{\mathrm{DD}} + 1$ distinct roots under the declared polynomial query cap. This *exceptional* root need not be uniform; the distributional deviation is already charged to the bad-branch budget. Such programming is consistent only for previously unqueried addresses; a prequery collision belongs to the separately budgeted DD-ROM exceptional event. Consequently the constructor guarantees exact algebraic recovery on every output under the delayed-polynomial game. Byte-level consistency additionally requires the declared ideal-XOF programming interface; the root-selection step is part of the exceptional constructor and its distributional cost is charged to the bad branch.

Exceptional-output limitation. The failure budget concerns the *joint* law averaged over outputs from the stipulated terminal experiment. For an arbitrarily rare prescribed o, the total-variation distance of this fallback from the exact conditional law can be close to one. No bound $\delta_{\mathrm{cond}}^{(1)}$ for *each fixed exceptional output* is claimed; the construction guarantees recovery there, while its distributional cost is paid in the average joint-law ledger.

Theorem 2 (Certified one-instance suffix kernel). *Suppose the certificate conditions in Subsections 7.4 and 7.5 hold uniformly over the declared profile, corrupted set, contributor supports, and good-residue domain. The exceptional mass is charged in* (81)*. Then there exists a uniform PPT suffix kernel whose error is at most* $\delta^{(1)}_{\mathrm{cond}}$ *in the* joint law averaged over the terminal output*, not in the conditional law of every rare fixed output. Every generated suffix nonetheless recovers the supplied raw output exactly:* $\mathsf{Recover}(\mathsf{suffix}) = O$.

Proof. The ideal kernel obtains (66) by Lemma 38. The convolution and coset reductions change the honest Gaussian joint law by the bounds in Lemmas 42 and 44. Lemma 43 limits the centered-lift substitution, and (75) gives the certified finite rejection budget. Inversion of the quantizers is exact when supplied their finite primitives; replaying the server's deterministic operations and its already-fixed arithmetic coins is also exact. The pivot-response interface and DD reparameterization preserve the prescribed response-code joint relation. The only remaining approximation is the certified finite-precision arithmetic. A coupling of all these replacements yields (81). The exceptional constructor of Lemma 45 gives exact algebraic recovery on the bad branches; its law may be far from the conditional law of a rare fixed output, but is paid for in the unconditional exceptional-mass ledger. All loops are covered by the uniform trial and precision caps, proving PPT time.

7.7 Late Distributed-Dither Reconstruction and Byte Programming

The previous construction concerns the polynomial-level terminal reference law. It remains to restore an actual public DD seed and ideal-XOF byte streams in the joint view. It would be incorrect to choose the dither seed *first*, condition on its complete already-observed streams, and then freely force those streams to match the desired output. This is why Hybrid H_1 deferred them.

Lemma 46 (DD reparameterization). *Fix all pre-DD partial-response values* $\mathbf{h} = (h_i)_{i\in\mathcal{C}}$*. Over a single released coefficient, let* $\mathsf{Off} \in \mathbb{Z}_Q$ *and* $D_i \in \mathbb{Z}_Q$ *be independent uniform public dithers, and put* $X_i = h_i + D_i + \mathbf{1}_{\{i=o^*\}}\mathsf{Off} \pmod{Q}$*. The joint distribution of* $(\mathsf{Off}, \mathbf{D}, \mathbf{X})$ *is identical to the distribution obtained by sampling* $\mathsf{Off}, \mathbf{X}$ *independently uniformly and setting*

$$D_i = X_i - h_i - \mathbf{1}_{\{i=o^*\}}\mathsf{Off} \pmod{Q}.$$

This identity holds coefficientwise and therefore for entire polynomials in R_Q*.*

Proof. For each fixed $\mathbf{h}$ and Off, the map $\mathbf{D} \mapsto \mathbf{X}$ is a coordinatewise affine bijection of $\mathbb{Z}_Q^{|\mathcal{C}|}$ with inverse given in the statement. Uniform measure is invariant under a bijection. Apply the identity to every independent coefficient and the prescribed domain-separated polynomial addresses.

Deferred ideal-XOF lifting. Let T be the number of completed rounds, $R_{\max}$ the certified maximum number of Sparse model groups, $R_a = \lceil d_{\mathrm{aux}}/N \rceil$, and ℓ_Q the RNS-limb count. A conservative count of DD residue requests is

$$C_{\mathrm{DD}} = NT(R_{\max} + R_a)(n+1)\ell_Q.$$

Suppose at most $q_X^{\mathrm{DD}} < 2^{128}$ classical ideal-XOF prequeries are allowed, the DD seed is a fresh 128-bit root per round, and a per-limb rejection routine is bounded by B_{DD}. If p_μ denotes the word-acceptance probability for RNS modulus μ, the bounded-oracle rejection interface specified in Section 7.7 gives the conservative ledger

$$\begin{aligned} \delta_{\mathrm{DD-ROM}} &\le \delta_{\mathrm{DD-guess}} + \delta_{\mathrm{DD-rej}}, \\ \delta_{\mathrm{DD-guess}} &\le \frac{T q_X^{\mathrm{DD}}}{2^{128} - q_X^{\mathrm{DD}}}, \\ \delta_{\mathrm{DD-rej}} &\le NT(R_{\max} + R_a)(n+1) \sum_{\mu \mid Q} (1-p_\mu)^{B_{\mathrm{DD}}}. \end{aligned} \tag{85}$$

For every target residue, the accepted byte words have equal-sized preimages under the rejection map. Conditional word sampling therefore produces precisely the required residue without changing the ideal-XOF law unless a prequery collision or capped rejection occurs. Base-mask XOF addresses are *not* reprogrammed in this step; the established setup and cache XOF table is retained.

Lemma 47 (Deferred-DD byte-level accounting). *Let P_{byte} be the real joint leakage/view law, P_{poly} the law with future DD values deferred as uniform polynomials, and Q_{poly} the corresponding computational terminal distribution. Let ϵ_0 bound the non-DD sampler, base-XOF, flooding and ssRLWE joint-law transitions between P_{poly} and Q_{poly}. If the certified terminal suffix kernel has overall error δ_{cond}, then the final output-preserving byte-level simulator can be accounted conservatively by*

$$\mathrm{Adv}^{\mathrm{whole}} \le 2\epsilon_0 + \delta_{\mathrm{cond}} + 3\delta_{\mathrm{DD-ROM}}. \tag{86}$$

The factor three is an explicit conservative charge for deferring the real DD bytes, transferring the permitted leakage marginal, and programming deferred polynomials back into byte streams.

Proof. The byte-to-polynomial deferred-decision experiment differs only when a DD address was prequeried or a bounded sampler fails, so (85) bounds its TV distance to the real experiment. The non-DD joint-law hybrids cost ϵ_0. Replacing the terminal conditional suffix by the certified PPT kernel costs δ_{cond}, and transferring from the terminal leakage marginal back to the real leakage marginal costs at most the original joint-law distance by data processing. Finally, for every as-yet-unqueried DD address the programmed byte stream is drawn conditionally on the already sampled residue; the same bad-event bound covers that lift. By the triangle inequality, charging the deferred and marginal transfers conservatively gives (86). A coefficient less than three would require a separate coupled-oracle analysis.

Canonical output bytes from the sampled suffix. The simulator emits the actual *quantizer codes*, not arbitrary integers that have the same Q-residue: the inverse kernels of Lemmas 34 and 35 return values in the stipulated finite code alphabets with their associated coins. The sorted Sparse label and group maps are already fixed by the semantic frontier. By Lemma 1, these values, the cached polynomial residues and the selected DD root can be serialized to the reference-wire representation without creating an alternative accepted encoding. Rejected or malformed bytes are not part of the honest protocol trace; the ideal-XOF programming and bounded-rejection exceptions remain charged in (85). Thus the suffix simulator's algebraic consistency and its canonical-wire consistency are separate, explicitly checked obligations.

7.8 Chronological Composition over Adaptive Training Rounds

The protocol reuses its setup and cached KSK/M2R material over the entire T-round run. Moreover, Sparse supports can depend on the previously released learning states. Consequently the per-instance posterior statement cannot simply be extrapolated by declaring all rounds independent. We use the application-causality condition of Section 3.4 and an explicit chronological factorization.

Definition 1 (Application/cryptographic causality firewall). *An admissible application policy may choose the round-t update, Sparse support, semantic trace, and local learning state as functions of its private application tape, public setup and the* past application-visible outputs and learning states. *The application policy cannot inspect the cryptographic engine's private random tapes, cached KSK/M2R bytes, ciphertext bytes, current or past quantizer coins, DD polynomial streams or other hidden transcript information. This restricts the causal input channel generating future application semantics; it does not remove those cryptographic objects from the server–colluder view.*

Four chronological distributions. For each round t, distinguish (a) the application transition π_t that chooses the semantic package A_t from the application-visible past, (b) the frontier law μ_t of Lemma 31, (c) the output likelihood ℓ_t induced by (62), and (d) the normalized suffix kernel K_t supplied by Theorem 2. Their order is

$$\underbrace{A_t}_{\pi_t} \longrightarrow \underbrace{\mathsf{Frontier}_t}_{\mu_t} \longrightarrow \underbrace{O_t}_{\ell_t} \longrightarrow \underbrace{\mathsf{Suffix}_t}_{K_t}.$$

The kernel π_t is *not* permitted to read a hidden past cryptographic state, whereas μ_t and K_t reuse that very state when forming the adversary's view. The relevant posterior fact is that ℓ_t does not depend on the numeric setup/frontier after the admitted semantic package and corrupted state are fixed (Lemma 33). We do not condition the current frontier on an arbitrarily chosen future support, because such a support can be a function of future application-visible output.

Theorem 3 (Chronological terminal posterior factorization). *Fix the setup cohort and corruption set, with the causality contract of Definition 1. Let*

$A_t = (M_t, \Phi_t^{\mathrm{sem}})$ be the supplied per-round semantic package and let O_t be the supplied raw output. In the deferred-DD terminal distribution, a uniform chronological generator exists that (i) samples one shared setup/cache state, (ii) for each t generates the data-free frontier from that same cache and the current public semantic layout, and (iii) generates the full conditional suffix from (A_t, O_t) and the generated history. It does not condition the current frontier on the cryptographic future or resample the setup after observing a later output.

Proof. Fix the external application tape and, for proof analysis only, the hidden index-permutation key. Factor the terminal execution in actual chronological order into an application transition, a frontier transition, an output likelihood and a normalized suffix transition. By Lemma 31, the frontier transition depends only on the current semantic layout, fresh base-mask inputs and setup/cache state, not on the unreleased honest c_1 suffix. By Lemma 33, the current output likelihood conditional on the semantic package does not depend on the numerical setup/frontier values. Therefore Bayes conditioning on O_t does not tilt the setup/frontier transition at round t. The application transition may depend on earlier M, O and on private application state, but by the causality firewall it cannot read the cryptographic randomness whose posterior we are sampling. Inducting on t yields the claimed factorization of the *cryptographic conditional kernel* while retaining the single shared cache. The hidden index key is irrelevant to that kernel once the supplied shuffled-label semantic layout is fixed. This is a chronological argument, not a claim that an arbitrary past frontier remains independent after conditioning on a future semantic trace generated with an unauthorized dependence on that frontier.

A uniform lifetime certificate. For Sparse, let

$$J = \lceil d/B \rceil, \quad Y_{\max} = \min\{J, \lfloor nK_\rho/\tau \rfloor\}, \quad L_m = BTY_{\max}, \quad L_a = NTR_a.$$

For Full take $L_m = NTR_m$, with $R_m = \lceil d/N \rceil$, and the same $L_a = NTR_a$. Let $\delta^{(1)}_{\mathrm{cond,m}}$ and $\delta^{(1)}_{\mathrm{cond,a}}$ be the model and auxiliary versions of (81). A conservative lifetime certificate for Sparse is

$$\begin{aligned} \delta^{\mathrm{SP}}_{\mathrm{cond}} \le L_m \max_{0\le a\le c} \max_{I_H \subseteq \mathcal{H}} \delta^{(1)}_{\mathrm{cond,m}}(n-a, a; I_H) \\ + L_a \max_{0\le a\le c} \delta^{(1)}_{\mathrm{cond,a}}(n-a, a), \end{aligned} \tag{87}$$

where the maximization is limited to semantically admissible sets and uses either the uniform Jensen envelope or an independently certified alternative. The Full expression uses $I_H = \mathcal{H}$ throughout. The union bound is valid with adaptively chosen supports and a cached key because every conditional per-instance error bound is valid uniformly over previous histories and every eligible current contributor set.

7.9 Prescribed-Leakage Distribution Transfer

The hybrid estimate (61) concerns the *joint* laws of leakage and view. It does not by itself imply that their conditionals are close for each rare leakage value. The

correct final step samples the *actual* permitted leakage marginal, then applies the terminal conditional transcript kernel.

Lemma 48 (PPT joint-to-conditional transfer). *Let $P_{\Lambda,V}$ be the real joint law and $Q_{\Lambda,V}$ a terminal joint law, with computational or statistical distinguishing distance at most ϵ. Suppose a uniform PPT kernel K produces, given λ drawn from Q_Λ, a view within δ_K of $Q_{V|\Lambda=\lambda}$ in average joint-law distance. Then $(\Lambda, K(\Lambda))$ with $\Lambda \leftarrow P_\Lambda$ is within $2\epsilon + \delta_K$ of $P_{\Lambda,V}$.*

Proof. Compare $P_{\Lambda,V}$ with $Q_{\Lambda,V}$ at cost ϵ. Replace the terminal conditional view by K at cost δ_K. Finally replace Q_Λ with P_Λ while retaining the same efficient kernel; by postprocessing/data processing this costs at most ϵ. This is a comparison of joint distributions, not pointwise conditioning on a leakage value of negligible probability.

Theorem 4 (Terminal simulator and conditioned transcript). *For fixed-cohort executions satisfying the security and randomness contracts of Sections 3–5 and every applicable hard TILT/coset/precision/ROM certificate above, there is a uniform PPT simulator $\mathsf{Sim}_{\mathrm{term}}$ that receives only the permitted leakage* (6) *and generates the terminal server–colluder transcript with average conditional statistical error at most δ_{cond}, as bounded by* (87) *for Sparse and its fixed-support Full specialization. All emitted model and auxiliary partial- decryption transcripts satisfy, without probability of failure,*

$$\mathsf{Recover}(\mathsf{Sim}_{\mathrm{term}}(\Lambda^{MO})) = O. \tag{88}$$

After delayed DD byte lifting, the resulting prescribed-leakage whole-view simulator has the conservative joint-distinguishing bound

$$\mathrm{Adv}^{\mathrm{whole}} \le 2\epsilon_0 + \delta_{\mathrm{cond}} + 3\delta_{\mathrm{DD-ROM}},$$

where ϵ_0 comprises the non-DD *sampler, base-XOF, flooding and ssRLWE hybrid costs. The complete profile-specific substitutions are given in Section 8.*

Proof. By Lemmas 31 and 33 and Theorem 3, sample the shared setup/cache once and run the terminal frontier kernel in the actual chronological order. Draw each certified suffix by Theorem 2, using the correct contributor set and path-specific parameters. This reconstructs all server computations and all partial-decryption codes; the per-coordinate deviations sum to δ_{cond} by (87). Reparameterize the deferred DD polynomials by Lemma 46 and lift them to ideal-XOF bytes under (85). The constructor enforces the reconstruction identity on every output coordinate, including its bounded exceptional branches, establishing (88). Finally apply Lemma 48 to the non-DD joint-law hybrids and charge the three DD byte/polynomial transitions from Lemma 47. At no point do we assume that a given exceptional leakage value has negligible conditional error; the guarantee is the stipulated whole-view joint-law advantage.

Scope of the result. The theorem is relative to the declared M, O and semantic metadata leakage; it does not assert security given only the released real-valued model. It assumes a classical programmable ideal-XOF, fixed corruption and cohort, the certified sampler budgets, the application causality firewall and the integer-lift conditions. It does not analyze operational dropout, survivor reconfiguration, retries, malicious clients or unauthorized application adaptation to cryptographic bytes. These distinctions are maintained in the three complete security theorems of the next section.

8 Complete Security Theorems for the Three Profiles

The preceding sections establish the two parts of the proof that must be composed: the whole-view game sequence of Section 6 and the prescribed-output conditional simulator of Section 7. The former was presented in full for Sparse with $k > 1$; the latter was constructed through a profile-dependent terminal kernel. We now discharge the profile-specific cases rather than treating Full or Sparse $k = 1$ as an unproved "analogous" application. We also distinguish the *conditional cryptographic theorem* from a concrete static parameter certificate. Throughout this section we speak about the fixed-cohort, no-dropout, no-retry Core protocol only.

8.1 Common Statement, Notation, and Proof Obligations

Fix one setup with n registered clients, a static set $\mathcal{A} \subseteq \mathcal{C}$ of at most c corrupted clients, $\mathcal{H} = \mathcal{C} \setminus \mathcal{A}$, and $h = |\mathcal{H}|$. The server is semi-honest and may collude with all clients in $\mathcal{A}$. Full and auxiliary releases require at least $\tau \geq c + 2$ registered participants; each released Sparse label requires at least τ contributors. In particular, each such aggregate contains at least two honest contributors after removal of corrupted contributions. This is a contribution-count requirement, *not* τ-out-of-n Shamir decryption: the collective-decryption request is answered by every member of the fixed cohort.

Let $\mathsf{P} \in \{\mathrm{F}, \mathrm{SP1}, \mathrm{SPk}\}$ denote, respectively, Full, Sparse with $k = 1$, and Sparse with $k > 1$. The experiment releases the aggregate and semantic information

$$\Lambda_{\mathsf{P}}^{MO} = \big(\mathsf{Cfg}, \boldsymbol{\alpha}, \Phi^{\mathrm{sem}}, \mathsf{CorSt}_{\mathcal{A}}, M, O\big) \tag{89}$$

from Section 3.3; the adversary's view includes every stipulated server cache, wire message, server random tape, and corrupted-client internal state. For a candidate simulator $\mathsf{Sim}_{\mathsf{P}}$ define its whole-view advantage against a PPT distinguisher $\mathcal{D}$ by

$$\begin{aligned}\mathrm{Adv}_{\mathsf{P}}^{\mathrm{whole}}(\mathcal{D}) := \big|\, &\Pr[\mathcal{D}(\Lambda_{\mathsf{P}}^{MO}, \mathsf{View}_{\mathcal{A}}^{\mathrm{real}}) = 1] \\ &- \Pr[\mathcal{D}(\Lambda_{\mathsf{P}}^{MO}, \mathsf{Sim}_{\mathsf{P}}(\Lambda_{\mathsf{P}}^{MO})) = 1]\big|.\end{aligned} \tag{90}$$

The two experiments use the *same real marginal law* of Λ^{MO}, as required by the simulation definition. No pointwise indistinguishability assertion for an arbitrarily rare leakage value is implicit in this definition.

Let

$$R_m = \left\lceil \frac{d}{N} \right\rceil, \qquad R_a = \left\lceil \frac{d_{\mathrm{aux}}}{N} \right\rceil, \qquad K_\rho = \max\left\{1, \left\lceil \frac{\rho d}{B} \right\rceil \right\}.$$

Write q_x^{P} for the maximum number of local ciphertext rows per honest encryption key over the entire T-round execution:

$$\begin{aligned} q_x^{\mathrm{F}} &= T(R_m + R_a), \\ q_x^{\mathrm{SP1}} &= T(K_\rho + R_a), \\ q_x^{\mathrm{SPk}} &= T(\lceil K_\rho/k \rceil + R_a). \end{aligned} \tag{91}$$

For example, the R_Q-ssRLWE term in the three computational theorems is respectively $h\epsilon_Q(q_x^{\mathrm{F}})$, $h\epsilon_Q(q_x^{\mathrm{SP1}})$ and $h\epsilon_Q(q_x^{\mathrm{SPk}})$. The two-sample KSK contribution is $2hm\epsilon_{QP}(2)$ in all cases; the extra $\epsilon_{QP}(km)$ appears only with M2R. These terms count the entire cached and online *sample families*, not the number of individual uses of a cached row. This is a worst-case row count, not a statement that all admissible Sparse labels are present in every round. All modes share the same fixed m KSK gadget limbs, T rounds, public seed/XOF conventions and code-sum collective decryption. The profile distinctions are summarized in Table 3.

The common released-phase equation, instantiated. To see precisely what differs between the three statements, consider one model output instance ν. The natural-error term to be flooded is *always* the integer lift

$$\Delta_\nu^{\mathbb{Z}} = E_\nu^{\mathrm{nat},0,\mathbb{Z}} + d_\nu^{\mathbb{Z}} z_{\mathcal{C},\nu}, \tag{92}$$

from (31), but its components follow Table 3: $z_{\mathcal{C},\nu}$ is $s_{\mathcal{C}}$ for Full and Sparse $k = 1$, whereas it is $t_{\mathcal{C}}$ for Sparse $k > 1$ model outputs. Only the last case includes the M2R error in $E_\nu^{\mathrm{nat},0,\mathbb{Z}}$. On *every* auxiliary block $z_{\mathcal{C},\nu} = s_{\mathcal{C}}$ and there is no M2R term. Thus (53) is a shared equation, while the bounds on $\mathbb{E}\sum_\nu \|P_\nu \Delta_\nu^{\mathbb{Z}}\|_2^2$ come from different propositions: 3 for Full and 2 for both Sparse regimes. After flooding, the terminal output has the one common form (62). In that equation the honest upload set $I_H(\nu)$ equals $\mathcal{H}$ for Full and auxiliary blocks, but can be a proper subset of $\mathcal{H}$ for a Sparse model label. The honest decryption responder count remains $h = |\mathcal{H}|$ in all three cases. This is why the same TILT/coset machinery has *different* model coefficients and energy certificates across the profiles.

Conditions shared by all three theorems. The statements below assume (i) independent small secrets in their specified domains and ssRLWE for R_Q and R_{QP}, (ii) a classical programmable ideal-XOF with explicitly bounded prequeries and rejection failures, (iii) the ideal/implemented sampler-discrepancy ledger, (iv) the fixed-cohort application causality contract of Definition 1, (v) a whole-lifetime Gaussian flooding certificate with an integer lift of every released phase, and (vi) uniformly certified polynomial-time TILT, Gaussian-coset, and

Table 3. Proof-relevant distinctions among the three Core profiles.

Operation or property	Full	Sparse $k = 1$	Sparse $k > 1$
Sparse supports	No	Yes	Yes
Selected b-lanes	No	Yes	Yes
R2M and M2R	No	No	Yes
Model decryption key	$s_{\mathcal{C}}$	$s_{\mathcal{C}}$	$t_{\mathcal{C}}$
HYBRID arithmetic	Deterministic	Stochastic	Stochastic
H5 M2R share	Identity	Identity	ssRLWE

finite-code inverse kernels on all admissible semantic histories. The last condition includes the exceptional-branch and finite-precision budgets of Section 7; it cannot be replaced by the observation that rejection sampling succeeds on a few example residues.

For P let ϵ_F^{P} denote the *one lifetime* flooding distance for model plus auxiliary outputs; let $\delta_{\mathrm{impl}}^{\mathsf{P}}$ and $\delta_{\mathrm{base-ROM}}^{\mathsf{P}}$ denote sampler and base-XOF costs, and let $\delta_{\mathrm{cond}}^{\mathsf{P}}$ and $\delta_{\mathrm{DD-ROM}}^{\mathsf{P}}$ denote the conditional simulator and late-DD byte-programming costs. All are nonnegative *whole-execution* quantities. Define

$$\begin{aligned}\epsilon_0^{\mathsf{P}}(h) := \epsilon_{\mathrm{comp}}^{\mathsf{P}}(h) + \epsilon_F^{\mathsf{P}} \\ + \delta_{\mathrm{impl}}^{\mathsf{P}} + \delta_{\mathrm{base-ROM}}^{\mathsf{P}}.\end{aligned} \tag{93}$$

The deferred-DD step is *not* hidden inside ϵ_0: its final coefficient is three, rather than two, as shown in Lemma 47.

8.2 Full: Direct Aggregation and Deterministic HYBRID

Theorem 5 (Full whole-view simulation security). *In the fixed-cohort Full Core with the common conditions above, there exists a uniform PPT simulator* $\mathsf{Sim}_{\mathrm{F}}(\Lambda_{\mathrm{F}}^{MO})$ *that reproduces the prescribed raw model and auxiliary outputs exactly and, for every PPT distinguisher, satisfies*

$$\begin{aligned}&\mathrm{Adv}_{\mathrm{F}}^{\mathrm{whole}} \le 2\epsilon_0^{\mathrm{F}}(h) + \delta_{\mathrm{cond}}^{\mathrm{F}} + 3\delta_{\mathrm{DD-ROM}}^{\mathrm{F}}, \\ &\epsilon_{\mathrm{comp}}^{\mathrm{F}}(h) := 2\epsilon_{QP}(1) + 2hm\epsilon_{QP}(2) + h\epsilon_Q\big(T(R_m + R_a)\big).\end{aligned} \tag{94}$$

Here ϵ_F^{F} *uses the* deterministic *Full error-energy analysis of Proposition 3, not the Sparse martingale bound. Recovery is exact:*

$$\mathsf{Recover}\big(\mathsf{Sim}_{\mathrm{F}}(\Lambda_{\mathrm{F}}^{MO})\big) = O.$$

Proof. We spell out each departure from the Sparse $k > 1$ game sequence. Set H_0^{F} to the true Full experiment, including every dense model block, auxiliary block, seed, setup cache, KSK use and compressed partial-decryption response. In H_1^{F} replace the public DD byte-generated polynomials by deferred uniform polynomials using precisely the setup-wide first-query and rejection exceptional-event ledger of Hybrid H_1. In H_2^{F} replace implementation samplers by their

stipulated ideal distributions; this changes the entire joint view by at most $\delta^{\mathrm{F}}_{\mathrm{impl}}$. In H^{F}_3 program the base ideal-XOF so that the common PK mask and local encryption masks can be sampled as their specified uniform polynomials before compatible seed streams are conditionally reconstructed. The change is bounded by $\delta^{\mathrm{F}}_{\mathrm{base-ROM}}$. No M2R masks are sampled in the Full path.

In H^{F}_4 use a fixed honest pivot $p^\star \in \mathcal{H}$ for each output instance ν. The Full aggregate phase satisfies the integer-lift form of Lemma 15. The pivot's Gaussian share is translated by the *complete* natural and aggregate-a compression error on the released N coordinates. The deterministic signed CRT digits and Mod-Down remainders are not conditionally centered, and their cross terms are not declared orthogonal. Instead, the Full-specific Minkowski estimate from Proposition 3 bounds the squared lifetime translation. Lemma 21 and Proposition 4 (the chronological KL chain rule) give a statistical distance no larger than ϵ^{F}_F. The same shared cache is reused for every decryption, and it is never replaced by fresh KSK rows merely to facilitate a per-round bound.

Full has no M2R share or third long-term key domain, so $H^{\mathrm{F}}_5 = H^{\mathrm{F}}_4$ exactly: its transition cost is zero. In H^{F}_6 insert a single R_{QP} ssRLWE challenge into the honest pivot's PK share, preserving the common target PK by summing the other shares under the same public mask p_1. The pivot's flooded partial response is generated through Lemma 24, without querying the unknown pivot secret. This costs $\epsilon_{QP}(1)$. In H^{F}_7 remove the $P\gamma_\lambda x_i$ payload from each of the hm honest KSK rows. The two-sample, forward-and-reverse ssRLWE transition is performed under a *single* shared PK target; all corrupted rows and subsequent key switches are consistently regenerated as in Lemmas 27–28. The cost is $2hm\epsilon_{QP}(2)$. In H^{F}_8 restore the pivot PK share via one reverse single-sample ssRLWE reduction at cost $\epsilon_{QP}(1)$. Since the honest KSK rows now have zero payload, rebuilding them under the restored target does not require knowledge of the honest x_i.

In H^{F}_9 replace the honest local model and auxiliary b-components by uniform R_Q components, without altering their independent local a masks, cryptographic seeds or the corrupted parties' prescribed states. The application fixes the next dense update before the next challenge row is revealed; hence a single adaptive $q^{\mathrm{F}}_x = T(R_m + R_a)$-row ssRLWE challenge under each honest local key suffices. The argument is the unprojected case of Lemma 30, and costs $h\epsilon_Q(q^{\mathrm{F}}_x)$. This verifies the claimed computational ledger and gives a joint law within $\delta^{\mathrm{F}}_{\mathrm{DD-ROM}} + \epsilon^{\mathrm{F}}_0$ of the true Full experiment.

It remains to simulate, not merely to define, the endpoint. Here the honest model-upload contributor set is $I_H(\nu) = \mathcal{H}$ for *every* dense output; so is the auxiliary contributor set. The Full terminal output identity follows from Lemma 32 with one finite upload error for each active honest source and one Gaussian/c_2 response from every honest decryption client. The Full-specific inverse code law and deterministic aggregate-a frontier satisfy Lemmas 34 and 31. Under its Full TILT/coset certificate, Theorem 2 generates the output-conditioned suffix, retaining the fixed cache and imposing $\mathsf{Recover} = O$ algebraically. Chronological factorization, conditional marginal transfer and late-DD byte lifting then

follow from Theorems 3 and 4. The terminal error is at most $\delta^{\mathrm{F}}_{\mathrm{cond}}$ over all $TN(R_m + R_a)$ released coefficient uses. Its byte-lifting accounting contributes the conservative $3\delta^{\mathrm{F}}_{\mathrm{DD-ROM}}$, whereas the non-DD hybrid cost occurs twice in the joint-to-conditional transfer. This proves the asserted bound and exact recovery. The proof is explicitly relative to Λ^{MO} and does not assert that the server learns only the final real-valued model update.

8.3 An Explicit Full/Single-Lane Sparse Game Interface

For clarity, the complete game variables for the two cases *without* M2R are specified here, rather than being identified by an unexplained analogy to the three-key-domain case. Let P^{P}_j denote the joint law of $(\Lambda^{MO}_{\mathsf{P}}, \mathsf{View})$ in game H^{P}_j. Each game retains the same registered clients, setup and exact cached-object identities as the corresponding real mode.

Table 4. Whole-view hybrid interfaces when M2R is absent. Each row changes the complete dependent transcript, not only one selected ciphertext.

Game	Full F	Sparse $k = 1$ SP1
H_0	Real dense blocks and fixed support	Real selected labels and partial uploads
H_1	Deferred DD polynomials	Deferred DD polynomials
H_2	Ideal reference samplers	Ideal reference samplers
H_3	Base-mask XOF coupling	Base-mask XOF coupling
H_4	Full deterministic residual flooded	Adaptive randomized-ZM residual flooded
H_5	Identity: no M2R	Identity: no M2R
H_6	Pivot s-PK share uniform	Pivot s-PK share uniform
H_7	All honest x-KSK payloads zero	All honest x-KSK payloads zero
H_8	Pivot PK law restored	Pivot PK law restored
H_9	All honest dense b rows uniform	All transmitted honest b projections uniform

Lemma 49 (Full and Sparse-$k = 1$ hybrid interface). *Under the assumptions of Theorems 5 and 6, the rows in Table 4 can be coupled using exactly the non-DD costs*

$$\begin{aligned}\epsilon^{\mathrm{F}}_{\mathrm{pre}} &= \delta^{\mathrm{F}}_{\mathrm{impl}} + \delta^{\mathrm{F}}_{\mathrm{base-ROM}} + \epsilon^{\mathrm{F}}_{F} + 2\epsilon_{QP}(1) + 2hm\epsilon_{QP}(2) + h\epsilon_{Q}(q^{\mathrm{F}}_{x}),\\ \epsilon^{\mathrm{SP1}}_{\mathrm{pre}} &= \delta^{\mathrm{SP1}}_{\mathrm{impl}} + \delta^{\mathrm{SP1}}_{\mathrm{base-ROM}} + \epsilon^{\mathrm{SP1}}_{F} + 2\epsilon_{QP}(1) + 2hm\epsilon_{QP}(2) + h\epsilon_{Q}(q^{\mathrm{SP1}}_{x}).\end{aligned} \tag{95}$$

The $H_4 \to H_5$ edge has zero *cost in both cases. The $H_3 \to H_4$ bounds use different certified energy estimates and must not be interchanged.*

Proof. The Full reference has R_m dense model blocks and R_a auxiliary blocks per round. Every local model or auxiliary ciphertext switches directly from x_i to

$s_{\mathcal{C}}$, so its complete aggregate has no M2R share, public M2R mask, or t_i domain. The deterministic CRT gadget/ModDown remainders are kept together in the Full natural-error lift; the L_2 triangle bound of Proposition 3 supplies its H4 cost.

For Sparse $k = 1$, $B = N$ and Split and Combine are identity maps; a local upload has at most one occupied label per row. An admitted label y is aggregated only from I_y, but its collective response is still given by the *entire* fixed cohort. The fresh randomized digit, stochastic ModDown, and unbiased aggregate-a coins are generated in the order certified by Lemmas 20 and 18. The cached rows are not regenerated on support changes. Proposition 2 therefore supplies its own H4 bound with the M2R energy set to zero. These facts establish $P_5 = P_4$ exactly in both cases, not just computationally.

For either mode the unknown secret after H6 is the pivot s_{p^*}; its flooded partial-decryption responses are generated without that secret by Lemma 24. The two-sample H7 reduction is used hm times and shares the same combined public-key target across cached rows. H8 reverses the PK replacement only after zeroing the payloads; H9 embeds a chronological local R_Q challenge under each x_i. In Full the output projection is the identity; in Sparse $k = 1$ it is the previously fixed admitted-support projection. Neither reducer chooses a future projection after seeing its next challenge row. Consequently the costs are the H6, H7, H8 and H9 terms written in (95), plus the mode-specific statistical costs of H2–H4. DD byte deferral is outside these two expressions and is charged separately.

8.4 Sparse with $k = 1$: Partial Uploads without M2R

Theorem 6 (Sparse $k = 1$ whole-view simulation security). *For the fixed-cohort Sparse Core with $k = 1$, the contributor threshold and all common certified conditions imply the existence of a uniform PPT simulator $\mathsf{Sim}_{\mathrm{SP1}}$ with exact recovery and*

$$\begin{aligned} &\mathrm{Adv}^{\mathrm{whole}}_{\mathrm{SP1}} \le 2\epsilon_0^{\mathrm{SP1}}(h) + \delta^{\mathrm{SP1}}_{\mathrm{cond}} + 3\delta^{\mathrm{SP1}}_{\mathrm{DD-ROM}}, \\ &\epsilon^{\mathrm{SP1}}_{\mathrm{comp}}(h) := 2\epsilon_{QP}(1) + 2hm\epsilon_{QP}(2) + h\epsilon_Q\big(T(K_\rho + R_a)\big). \end{aligned} \tag{96}$$

Unlike Full, its flooding and suffix conditions must be valid uniformly over adaptive Sparse supports.

Proof. For each training attempt the client selects at most K_ρ chunks and reveals permitted shuffled label and admission data before encrypting the selected updates. Because $k = 1$ and $B = N$, each selected chunk occupies one full RLWE polynomial. The server still performs label-wise weighted aggregation and responds only on admitted labels. In particular, the Sparse contributor set for a label need not equal the full cohort, while the decryption responder cohort remains $\mathcal{C}$.

Define H_0^{SP1} by the complete true fixed-cohort Sparse $k = 1$ execution. The first three games defer DD bytes, idealize implementation samplers, and couple

the base XOF to independent uniform masks in exactly the chronological order specified in Section 6.2. There is no M2R CRS or t_i key in this mode. The relevant public key is $s_{\mathcal{C}}$, and model and auxiliary decryption both use s_i shares. The base-ROM ledger includes common PK masks and seeded local upload masks; it does not include fictitious M2R masks.

For H_4^{SP1}, let $I_\nu \subseteq \mathcal{C}$ be the admitted contributors of label ν. The selection indicator for each source and coordinate is fixed before the fresh signed digit, stochastic ModDown and unbiased aggregate-a coins are exposed. The already-generated KSK row errors remain cached, and the support may be correlated with earlier outputs and those cached errors. Apply Lemma 18 *pathwise* and Lemma 17 to the newly centered arithmetic increments. This is not a Full deterministic HYBRID bound, nor does it assume support-independent cached keys. Since $k = 1$, the M2R term in (42) is identically zero. The model and auxiliary natural errors, plus the aggregate-a compression error multiplied by $s_{\mathcal{C}}$, are shifted into the honest pivot Gaussian on released coordinates. By Proposition 2 and the chronological KL bound of Proposition 4, the $H_3 \to H_4$ distance is at most $\epsilon_F^{\mathrm{SP1}}$.

Define $H_5^{\mathrm{SP1}} = H_4^{\mathrm{SP1}}$; there are no M2R shares to uniformize. The $H_5 \to H_6$ reduction first changes the pivot's s-PK share to a uniform polynomial. The flooded pivot model and auxiliary responses are reconstructed without the pivot's s key, exactly as in Lemma 24. The $H_6 \to H_7$ transition removes all honest client KSK payloads via the shared two-sample ssRLWE embedding; the $H_7 \to H_8$ transition restores the pivot PK under zero-payload KSK rows. Their costs, respectively, are $\epsilon_{QP}(1)$, $2hm\epsilon_{QP}(2)$ and $\epsilon_{QP}(1)$. With the target secret no longer linked to an honest x_i payload, one R_Q challenge per honest local key replaces its entire adaptively ordered family of local uploads, at most $q_x^{\mathrm{SP1}} = T(K_\rho + R_a)$ rows. The ciphertext b components become uniform on actually transmitted positions in the terminal game, costing $h\epsilon_Q(q_x^{\mathrm{SP1}})$. Every future support is generated by the stipulated application policy from past permitted outputs, not from the next challenge sample. Thus the adaptive projection argument of Lemma 30 applies without independence assumptions about the *past*.

At the endpoint, for each released label y, the honest upload-error sum comprises the sources in $I_H(y) = I_y \cap \mathcal{H}$, while *every* honest member of $\mathcal{H}$ provides a Gaussian and c_2 response. The label-wise admitted-support metadata is part of Φ^{sem} and the aggregated ideal M_y is permitted leakage. For every admissible $I_H(y)$, the finite quantization law, Gaussian convolution, coset splitter and TILT acceptance certificate of Section 7 yield a uniform suffix kernel. It uses neither the original secret permutation nor individual honest updates. The support-dependent maximum in (87) covers the entire $L_m = BTY_{\max}$ model-coordinate lifetime, and the dense auxiliary path has $L_a = NTR_a$ coefficient uses. A single chronological cache/frontier generator and the prescribed-leakage transfer complete the proof. The TILT exceptional branches are included in $\delta_{\mathrm{cond}}^{\mathrm{SP1}}$, and the late-DD cost is separately counted three times. The suffix enforces $\mathsf{Recover} = O$ for every simulated output, including exceptional branches.

8.5 Sparse with $k > 1$: Three Key Domains and M2R

Theorem 7 (Sparse $k > 1$ whole-view simulation security). *Under the same fixed-cohort conditions for Sparse with $k > 1$, there is a uniform PPT simulator $\mathsf{Sim}_{\mathrm{SPk}}$ with exact recovery and*

$$\begin{aligned} &\mathrm{Adv}^{\mathrm{whole}}_{\mathrm{SPk}} \le 2\epsilon_0^{\mathrm{SPk}}(h) + \delta^{\mathrm{SPk}}_{\mathrm{cond}} + 3\delta^{\mathrm{SPk}}_{\mathrm{DD-ROM}}, \\ &\epsilon^{\mathrm{SPk}}_{\mathrm{comp}}(h) := \epsilon_{QP}(km) + 2\epsilon_{QP}(1) \\ &\qquad\qquad + 2hm\epsilon_{QP}(2) + h\epsilon_Q\big(T(\lceil K_\rho/k\rceil + R_a)\big). \end{aligned} \tag{97}$$

In this case the model output uses $t_{\mathcal{C}}$, while the auxiliary path uses $s_{\mathcal{C}}$.

Proof. The exact H0–H9 argument for this regime is Section 6; here we verify how it composes with the terminal kernel and with all profile-specific error budgets. Every local model ciphertext packs at most k admitted chunks, possibly with different source-specific support layouts. R2M separates the occupied interleaved lanes; Lemma 6 proves the negacyclic polynomial identity and Lemma 8 proves that zero-filling omitted b coefficients does not change an occupied lane's plaintext equation. The server aggregates by admitted shuffled label, then M2R key-switches and repacks up to k label aggregates under $t_{\mathcal{C}}$. The auxiliary path remains dense under $s_{\mathcal{C}}$. Each of these relations is an exact modulo-Q identity before introducing its error lift.

The H1–H3 transformations use one consistent byte/XOF table, actual fixed setup cache, and properly domain-separated seed material. The H4 Gaussian translation removes the complete model natural error (including both direct KSK and M2R) and aggregate-a compression times $t_{\mathcal{C}}$, and removes the corresponding direct-path and aggregate-a errors times $s_{\mathcal{C}}$ on auxiliary outputs. Proposition 2 includes the $k > 1$ M2R energy; its use of predictable selection, fresh arithmetic coins and a pathwise incidence bound accommodates adaptive support and cached keys. The pivot Gaussian is shifted only at released positions and only after the corresponding lift is fixed, so Proposition 4 supplies $\epsilon_F^{\mathrm{SPk}}$ without a roundwise-independence assumption.

The ordering of computational games matters. In H_5, the km cached M2R b shares of one honest pivot are replaced with uniform R_{QP} polynomials, while retaining their shared public a components. Only after this step can H_6 uniformize that pivot's public s-key share without leaving a known affine relation involving its unknown s_i in the unremoved M2R cache. H_7 removes the honest local x_i KSK payloads using the two-sample reduction for each row; H_8 restores the pivot public-key share under the new zero-payload cache. Finally H_9 replaces each honest local transmitted b projection by a uniform projection, using at most $T(\lceil K_\rho/k\rceil + R_a)$ adaptive challenge rows per honest source. The costs are, respectively,

$$\epsilon_{QP}(km), \quad \epsilon_{QP}(1), \quad 2hm\epsilon_{QP}(2), \quad \epsilon_{QP}(1), \quad h\epsilon_Q(q_x^{\mathrm{SPk}}),$$

which yield the computational expression in (97). All dependent server arithmetic and all partial-decryption messages are recomputed under the replacement cache; the same cached row is never treated as fresh on later use.

Once H9 has eliminated the local honest plaintext contents, the output-conditioned terminal law has the normal form of Lemma 32. The honest c_1 term depends on the actual admitted label contributors $I_H(y)$; the Gaussian and response-code terms contain all h honest collective-decryption clients. The frontiers contain the M2R A columns and stochastic arithmetic coins but not the honest c_1 suffix, and so obey the posterior-sufficiency statement of Lemma 33. The certified A_{h-1} coset splitter is applied to the h honest Gaussian shares independently of how many honest clients uploaded a particular Sparse label. Since the TILT/coset bound is uniform over the permitted contributor sets and semantic histories, chronological composition costs at most $\delta_{\mathrm{cond}}^{\mathrm{SPk}}$. Late DD programming yields the prescribed raw output for every released coefficient and contributes the final conservative $3\delta_{\mathrm{DD-ROM}}^{\mathrm{SPk}}$ term. The prescribed-leakage transfer of Lemma 48 charges non-DD hybrid distance twice, giving the claimed inequality. No statement is made for survivor reconfiguration or retry branches.

8.6 A Uniform Ledger and the Profile-Specific Conditions

For quick comparison, the computational part of the three theorems is

$$\begin{aligned}
\epsilon_{\mathrm{comp}}^{\mathrm{F}}(h) &= 2\epsilon_{QP}(1) + 2hm\epsilon_{QP}(2) + h\epsilon_Q\big(T(R_m + R_a)\big),\\
\epsilon_{\mathrm{comp}}^{\mathrm{SP1}}(h) &= 2\epsilon_{QP}(1) + 2hm\epsilon_{QP}(2) + h\epsilon_Q\big(T(K_\rho + R_a)\big),\\
\epsilon_{\mathrm{comp}}^{\mathrm{SPk}}(h) &= \epsilon_{QP}(km) + 2\epsilon_{QP}(1) + 2hm\epsilon_{QP}(2)\\
&\quad + h\epsilon_Q\big(T(\lceil K_\rho/k\rceil + R_a)\big).
\end{aligned} \tag{98}$$

Correct flooding scope. For Full, use

$$\epsilon_F^{\mathrm{F}} \le \frac{1}{2}\sqrt{TR_m\frac{(R_m^F)^2}{\sigma_{\mathrm{smg}}^2} + TR_a\frac{(R_a^F)^2}{(\sigma_{\mathrm{smg}}^{\mathrm{aux}})^2}}, \tag{99}$$

where R_m^F, R_a^F are defined in Proposition 3. For Sparse $k = 1$ or $k > 1$, let $\mathcal{E}_m^{F,\mathrm{SP}}$ and $\mathcal{E}_a^{F,\mathrm{SP}}$ be the respective bounds in Proposition 2 with the M2R term present only when $k > 1$. Then

$$\epsilon_F^{\mathrm{SP}} \le \frac{1}{2}\sqrt{\frac{\mathcal{E}_m^{F,\mathrm{SP}}}{\sigma_{\mathrm{smg}}^2} + \frac{\mathcal{E}_a^{F,\mathrm{SP}}}{(\sigma_{\mathrm{smg}}^{\mathrm{aux}})^2}}. \tag{100}$$

These are bounds for all released outputs over all T rounds, not per-round estimates. Full's noncentered deterministic ModDown remainders are controlled through L_2 triangle bounds; Sparse's conditional centering and staged orthogonality are controlled through its specified fresh-coin filtration. The two analyses are not interchangeable.

Correct conditional-kernel scope. For Full, the contributor set is fixed and its conditional error bound is multiplied by $TN(R_m + R_a)$ released coefficient uses. For Sparse, define $Y_{\max}$ and L_m, L_a as in (87), and take the uniform worst case

of $\delta^{(1)}_{\mathrm{cond,m}}$ over each admissible honest contributor set and corruption pattern. The honest c_2 and Gaussian response count remains h regardless of the number of uploaded chunks; otherwise the conditional Gaussian convolution and A_{h-1} coset would be parameterized incorrectly. The protocol's integer-lift, no-wrap and no-clipping requirements are *separate numerical correctness* conditions; the security comparison is not conditioned on the success of an entire realized training trajectory.

Static worst case. When one certificate must cover every admissible static corrupt set of size at most c, it may replace h by n in the monotone computational ledger and maximize the certified conditional errors over every admissible (a, I_H). The DD- and base-XOF query budgets, sampler calls, T-round arithmetic energy and all finite-trial caps must be evaluated over the same candidate and same total execution horizon. Charging a cached M2R key T times is unnecessarily pessimistic, whereas treating a reused KSK row as fresh at each round invalidates an independence argument.

8.7 Target-Specific Secret-Key Recovery Bounds

Whole-view transcript simulation is distinct from recovery of the full secret x_i, s_i, or (when $k > 1$) t_i. To establish a *key or polynomial-size candidate-list recovery* result we must stop a hybrid at the *targeted* secret and use a test whose error is explicit. The bounds in this subsection concern entire-key recovery, not arbitrary single-bit leakage. Fix an honest target i and rerun the hybrid proof with $p^\star = i$.

Lemma 50 (Held-out ssRLWE test for a target key). *Let $(a^\dagger, Z^\dagger)$ be one additional,* withheld *ssRLWE challenge sample under a secret z in the ring $R_{\mathcal{Q}}$, where $\mathcal{Q} \in \{Q, QP\}$. This sample is not provided to the key-list adversary. Suppose $\Pr[\|e\|_\infty > B_z] \le \delta_{z,\mathrm{test}}$, $2B_z + 1 < \mathcal{Q}$, and the adversary outputs at most ℓ candidates based on the protocol view. Then*

$$\Pr[z \in L] \le \epsilon^{(z)}_{\mathrm{endpoint,test}} + \delta_{z,\mathrm{test}} + \ell \left(\frac{2B_z + 1}{\mathcal{Q}} \right)^N . \tag{101}$$

The endpoint quantity covers the entire *targeted view transition, including the extra challenge sample, conditioned suffix and ROM/statistical costs.*

Proof. Given a candidate z', the external reducer tests $\|\mathsf{Center}_{\mathcal{Q}}(Z^\dagger + a^\dagger z')\|_\infty \le B_z$. In the structured challenge branch, the true z passes except on the certified error-tail event. In the fully uniform challenge branch, the withheld $Z^\dagger$ is independent of the adversary's view, its output list and $a^\dagger$. For *each* fixed list member the tested residue is uniform on $R_{\mathcal{Q}}$; the chance of the centered coefficient box is $((2B_z + 1)/\mathcal{Q})^N$. A union bound over the at most ℓ members completes the test. The test is run by the reduction, which does not ask the adversary whether its unknown challenge secret is in the list.

Uniform target-pivot cost envelope. For every honest target i, introduce five certified, full-lifetime budgets:

$$F_i, \qquad I_i, \qquad R_i^{\mathrm{base}}, \qquad C_i^{\mathrm{cond}}, \qquad R_i^{\mathrm{DD}}.$$

They bound flooding, sampler, base-XOF, conditional-suffix and late-DD costs, respectively. These are *target-specific* budgets, and the published worst-case certificate may be used only if it holds uniformly for all choices $p^\star = i$. Define

$$\mathcal{B}_i(C) := 2\big(C + F_i + I_i + R_i^{\mathrm{base}}\big) + C_i^{\mathrm{cond}} + R_i^{\mathrm{DD}}. \tag{102}$$

The factor two is the conservative joint-to-prescribed-leakage transfer; the DD expense is not set to zero or incorporated in an ordinary ssRLWE sample count. Under the uniform, pivot-independent static bounds of Section 9.2, this is bounded further by

$$\mathcal{B}_i(C) \le 2\big(C + \epsilon_F^{\mathrm{cert}} + \delta_{\mathrm{impl}}^{\mathrm{cert}} + \delta_{\mathrm{base-ROM}}^{\mathrm{cert}}\big) + \delta_{\mathrm{cond}}^{\mathrm{cert}} + 3\delta_{\mathrm{DD-ROM}}^{\mathrm{cert}}. \tag{103}$$

Here the last conservative expression uses the three-charge DD bound of Theorem 4; its right-hand side should not be mixed with an independently uncertified, target-optimized error ledger.

Theorem 8 (Explicit target-key computational ledger). *For profile* $\mathsf{P} \in \{\mathrm{F}, \mathrm{SP1}, \mathrm{SPk}\}$*, let* q_x^{P} *be as in* (91)*. In the order in which targeted games are* stopped*, the computational budgets are*

$$\begin{aligned} C_{s,i}^{\mathsf{P}} &= \mathbf{1}_{\mathsf{P}=\mathrm{SPk}}\epsilon_{QP}(km) + \epsilon_{QP}(1), \\ C_{x,i}^{\mathsf{P}} &= \mathbf{1}_{\mathsf{P}=\mathrm{SPk}}\epsilon_{QP}(km) + 2\epsilon_{QP}(1) + 2m\epsilon_{QP}(2) + \epsilon_Q(q_x^{\mathsf{P}} + 1), \\ C_{t,i}^{\mathrm{SPk}} &= \epsilon_{QP}(km+1). \end{aligned} \tag{104}$$

Here $C_{t,i}$ *is defined only for Sparse* $k > 1$*. The exact targeted endpoint terms satisfy the conservative certificates*

$$\begin{aligned} \epsilon_{\mathrm{endpoint}}^{(s_i)} &\le \mathcal{B}_i(C_{s,i}^{\mathsf{P}}), \\ \epsilon_{\mathrm{endpoint,test}}^{(x_i)} &\le \mathcal{B}_i(C_{x,i}^{\mathsf{P}}), \\ \epsilon_{\mathrm{endpoint,test}}^{(t_i)} &\le \mathcal{B}_i(C_{t,i}^{\mathrm{SPk}}) \quad (\mathsf{P} = \mathrm{SPk}). \end{aligned} \tag{105}$$

These are separate from the all-sources *whole-view simulation cost in* (98)*.*

Proof. For t_i with $k > 1$, run H1–H4 with model and auxiliary errors flooded and then stop at the pivot M2R transition H5. The target's km M2R rows and one withheld test row are embedded together under its unknown t_i, accounting for $\epsilon_{QP}(km+1)$. The target s_i case uses the full model (including direct KSwitch s-dependent errors) and auxiliary flooding, first removes the pivot M2R share if $k > 1$, and stops at the uniform pivot PK H6. Its computational cost is the M2R term, when present, plus one PK challenge sample.

For x_i, proceed through the target's H6 PK randomization, the target's m KSK payload removals using forward and reverse two-sample replacements, and H8 PK restoration. Unlike the ordinary whole-view game, other honest sources need not have their own payloads removed to conduct this *targeted* experiment: the reducer samples those known secrets and keeps all their original dependent cached rows. After the target payload is removed, its q_x^{P} chronologically admissible local rows and the one withheld sample form one R_Q ssRLWE family, costing $\epsilon_Q(q_x^{\mathsf{P}}+1)$. All involved target-dependent server arithmetic and corrupted/client shares are replayed under one common PK and KSK target; no cached row is replaced by an independent fresh row on later uses.

The conditioned-view comparison uses the same prescribed (M, O) law and terminal kernel as Lemma 48; its generic conservative cost is $\mathcal{B}_i$ including each statistical expense. For Full or Sparse $k = 1$ the M2R challenge is absent, not a hidden instance of $\epsilon_{QP}(km)$. This gives (104)–(105).

Corollary 1 (Complete-key/list recovery for each domain). *Suppose L has at most ℓ candidates and the target small-secret domains are generated as specified at setup. With $B_x, B_t, B_{\mathrm{pk,i}}$ certified from the corresponding error tails, the following profile-dependent upper bounds hold:*

$$\Pr[x_i \in L] \le \mathcal{B}_i(C_{x,i}^{\mathsf{P}}) + \delta_{x,\mathrm{test}} + \ell \left(\frac{2B_x+1}{Q}\right)^N, \tag{106}$$

$$\Pr[t_i \in L] \le \mathcal{B}_i(C_{t,i}^{\mathrm{SPk}}) + \delta_{t,\mathrm{test}} + \ell \left(\frac{2B_t+1}{QP}\right)^N \quad (\mathsf{P} = \mathrm{SPk}), \tag{107}$$

$$\Pr[s_i \in L] \le \mathcal{B}_i(C_{s,i}^{\mathsf{P}}) + \delta_{\mathrm{pk,i}} + \min\left\{1, \left(\frac{3(2B_{\mathrm{pk,i}}+1)}{QP}\right)^N\right\}. \tag{108}$$

For the s_i test the entire ternary candidate domain is union-bounded; a mere ℓ-factor is not *used in place of this bound because the public PK test itself is visible to the adversary. Each testing-failure budget is charged once, outside $\mathcal{B}_i$.*

Proof. Apply Lemma 50 to the x_i and t_i withheld samples and Theorem 8 for their targeted endpoint transitions. For s_i, the public anchor test is $\mathbf{1}\{s' \in \{-1,0,1\}^N\}\mathbf{1}\{\|\mathsf{Center}_{QP}(p_{0,i} + p_1 s')\|_\infty \le B_{\mathrm{pk,i}}\}$. A genuine s_i passes when its error f_i is within the certified bound. At a uniform pivot-PK endpoint the probability that *any* ternary candidate passes is at most $3^N((2B_{\mathrm{pk,i}}+1)/(QP))^N$, regardless of the adversary's chosen list. The claimed bounds follow by the same targeted whole-view transfer without inspecting any challenge secret.

8.8 Numerical Reference Configuration: Scope of the Claim

The Full and Sparse security theorems above are conditional results: they require cryptographic assumptions and the stated statistical, sampling, correctness, and execution-scope certificates. Section 9.4 specifies one *fixed-cohort* Full reference

candidate and one Sparse reference candidate with $k = 4$; their reported numerical bounds appear in Table 7. This is evidence for these two exact candidates, not a new reduction or a numerical certificate for Sparse $k = 1$.

For an exact candidate satisfying its complete numerical/reference checks, the general theorem yields the conditional conclusion

$$\mathrm{Adv}_{\mathsf{P}}^{\mathrm{whole}} \leq 2\epsilon_{\mathrm{comp}}^{\mathsf{P}} + 2^{-18}. \tag{109}$$

The remaining $\epsilon_{\mathrm{comp}}^{\mathsf{P}}$ is a *symbolic ssRLWE distinguishing-advantage term*, not a numerically certified value of zero or 2^{-128}. Reference-model static validation also does not establish security for dropout/retry executions or certify a deployed binary.

Concrete-seed versus asymptotic interpretation. The representative Core fixes its public seeds at 128 bits, so the prequery collision expressions in (85) and the base-XOF ledger are *concrete* finite-security bounds. A fixed value 2^{128} is not by itself an asymptotically negligible function of a variable security parameter λ. If one wishes to interpret the above games as a family of asymptotic uniform-PPT experiments, the setup must expose a seed-length function $\ell_{\mathrm{seed}}(\lambda)$, a polynomial query budget and polynomial implementations of the certified arithmetic/sampling interfaces, and require $q_{\mathrm{seed}}(\lambda)q_X(\lambda)/(2^{\ell_{\mathrm{seed}}(\lambda)} - q_X(\lambda)) = \mathrm{negl}(\lambda)$. The reported 128-bit candidate is instead judged under its concrete static validation policy.

8.9 Correctness Scope and Final Security Summary

Algebraic versus numerical correctness. All three profiles have the exact phase and repacking identities from Section 5 modulo Q. To identify the delivered centered result with the intended integer-reference update requires the separate input bounds, numerical precision, no-wrap and final signed-code no-clipping conditions from Section 5.5. The reference RMS statements do not by themselves bound the unconditional actual modular-output RMS on all exceptional executions. The correctness failure budget δ_{corr} belongs to this numerical contract, not to the privacy advantage sum; the security games themselves are not globally conditioned on a no-wrap event.

Exact boundary of what has been proved. The preceding theorems cover one setup, a fixed registered and decrypting cohort, static corruption, semi-honest parties, permitted M, O and semantic leakage, classically programmable ideal-XOF, certified samplers and conditional kernels, and the application-causality restriction. Survivor-dependent public keys, changing KSK/M2R targets, dropout and retry transcripts, and malicious behavior are separate operational features and are *not* consequences of the fixed-cohort proof. Likewise, label shuffling does not promise original-index secrecy when a colluding client knows the common permutation key.

Result. For all three Core modes, the complete proof consists of: exact algorithmic semantics (Sections 2–4), arithmetic and statistical lemmas (Section 5), whole-view H0–H9 transformations (Section 6 and this section), and a uniform prescribed-leakage terminal kernel (Section 7). The profile-specific bounds are (94), (96), and (97). These conclude simulation relative to the stipulated leakage and exact prescribed-output recovery, under the stated assumptions and uniform certificates; they do not replace the actual numerical audit of any candidate parameter set.

9 Parameter Conditions and Communication Analysis

This section connects the security and correctness conditions proved in Sections 5–8 to concrete protocol parameters and to the communication payloads of Section 4. Three levels of statement must be kept distinct. First, the whole-view security theorems are *conditional* results under the declared ssRLWE, sampler, ideal-XOF, execution, and application-causality assumptions. Second, a *static parameter certificate* checks those conditions for one particular mathematical reference configuration and bounded execution. Third, an experimental deployment may introduce networking, software, and scheduling effects that are not certified by the first two statements. The two concrete reference configurations below have reported static-validation results. We reproduce their parameters and numerical upper bounds, but do not claim to have rerun the underlying interval-arithmetic certification programs.

9.1 Admissible Parameters and Security Requirements

Cryptographic parameter domain. Let N be a power of two, $Q = \prod_{\lambda=0}^{m-1} q_\lambda$, and let P be the special key-switching modulus. The pairwise distinct ring primes and their CRT/NTT compatibility must be checked for the actual parameter tuple, rather than inferred from their bit lengths. Sparse further requires $N = kB$ with $k \mid N$. The fixed cohort has n registered clients, with a static corrupted set of size at most c and a minimum released-contributor threshold $\tau \geq c + 2$. The precise conditions on released Sparse labels and on auxiliary outputs are those of Section 3.

For the representative family examined below, $N = 8192$, $m = 2$, small secrets are independent uniform ternary polynomials, and the base encryption/key-generation error width is $\sigma_e = 3.19$. For the reference configurations, the classical Category-128 security-table check is applied to $N = 8192$, uniform ternary secrets and base error width $\sigma_e = 3.19$. The check compares the actual integer QP with 2^{214}, subject to these specified distribution, dimension and attack-model requirements. This check is not, by itself, an upper bound on the decisional ssRLWE advantage appearing in our hybrid games. In particular, an estimated attack-cost category must not be substituted for a concrete equality such as $\epsilon_{\mathrm{comp}} = 2^{-128}$.

Whole-execution noncomputational budget. For each profile P let ϵ_F^{P} be the lifetime Gaussian-translation cost across the model and auxiliary paths, let $\delta_{\mathrm{impl}}^{\mathsf{P}}$ measure sampler discrepancies, $\delta_{\mathrm{cond}}^{\mathsf{P}}$ the terminal conditional-kernel error, and $\delta_{\mathrm{base-ROM}}^{\mathsf{P}}$ and $\delta_{\mathrm{DD-ROM}}^{\mathsf{P}}$ the two XOF-related costs. These are not per-round bounds: each already accounts for all required calls over the fixed T-round execution. The reference validation policy requires

$$\begin{aligned} \epsilon_F^{\mathsf{P}} &\le 2^{-20}, \\ 2\epsilon_F^{\mathsf{P}} + 2\delta_{\mathrm{impl}}^{\mathsf{P}} + \delta_{\mathrm{cond}}^{\mathsf{P}} + 2\delta_{\mathrm{base-ROM}}^{\mathsf{P}} + 3\delta_{\mathrm{DD-ROM}}^{\mathsf{P}} &\le 2^{-18}. \end{aligned} \tag{110}$$

The factors of two and three follow the two-sided prescribed-leakage transfer and the late-DD accounting in Section 7; they are not additional independent events. Under the symbolic ssRLWE costs from Section 8.6, a candidate meeting (110) obtains the conditional consequence

$$\mathrm{Adv}_{\mathsf{P}}^{\mathrm{whole}} \le 2\epsilon_{\mathrm{comp}}^{\mathsf{P}} + 2^{-18}, \tag{111}$$

with $\epsilon_{\mathrm{comp}}^{\mathsf{P}}$ *not* numerically certified by the static security-table lookup.

Numerical correctness and its separate failure budget. Write $\ell \in \{m, a\}$ for the model and auxiliary paths, $W_1 = 128$ for the fixed integer weight sum, Δ_ℓ for the encoding scale, $\Delta_{d,\ell}$ for the delivery scale, and p_ℓ for the target relative RMS precision. A representative correctness gate is

$$S_{\mathrm{final},\ell}^{\mathrm{cert}} \le U_{\max,\ell} 2^{-p_\ell}, \tag{112}$$

where the certified reference-output bound takes the form

$$S_{\mathrm{final},\ell}^{\mathrm{cert}} = \frac{S_{\mathrm{int},\ell}^{\mathrm{cert}}}{W_1 \Delta_\ell} + \frac{1}{2\Delta_{d,\ell}}. \tag{113}$$

The intermediate integer-error radius $S_{\mathrm{int},\ell}^{\mathrm{cert}}$ is computed by the profile-specific energy analysis; it must not be replaced by a screening variance approximation. The actual integer representatives must additionally satisfy the no-wrap and signed final-code no-clipping inequalities established for the reference protocol. These are stronger than simply requiring the ideal scaled update to fit in its output code. For each path $\ell \in \{m, a\}$, the reference-model sufficient conditions are

$$\begin{aligned} B_{M,\ell} &:= W_1(\Delta_\ell U_{\mathrm{in},\ell} + \tfrac{1}{2}), \\ B_{M,\ell} + B_{\mathrm{num},\ell} &< Q/2, \\ \Delta_{d,\ell}\Big(U_{\mathrm{in},\ell} + \frac{B_{\mathrm{num},\ell} + W_1/2}{W_1 \Delta_\ell}\Big) + \tfrac{1}{2} &\le 2^{c_{d,\ell}-1} - 1. \end{aligned} \tag{114}$$

Here $U_{\mathrm{in},\ell}$ is an *a priori* bound on all admissible inputs on that path, including Sparse residual updates where applicable; $B_{\mathrm{num},\ell}$ is the separately certified lifetime integer-error *tail* bound excluding the displayed plaintext encoding allowance. It is composed of the applicable local encryption, c_1, KSK, optional

M2R, aggregate-a, smudging and DD terms, using each profile's actual sampling and cache/norm conditions. The $W_1/2$ contribution in $B_{M,\ell}$ is the sum of plaintext-encoding half-unit errors, and the corresponding term in the delivery gate accounts for the same input encoding; it is not an extra independent privacy loss. For a signed $c_{d,\ell}$-bit result the positive representable bound is $2^{c_{d,\ell}-1}-1$. Both inequalities must hold on *each* model/auxiliary path, together with the specified tail-failure allocation.

The static-validation policy requires the probability of *any* numerical correctness failure during the planned execution to be at most

$$\delta_{\mathrm{corr}} \le 2^{-30}. \tag{115}$$

This failure probability is separate from (110); in particular, the privacy games are not conditioned on a global no-wrap event. A bound on the *integer reference* RMS does not automatically imply an unconditional RMS bound on actual modular outputs when wrap, cache-good, or clipping conditions fail. For Sparse, the stated reference RMS guarantee must retain its setup-good or norm-good conditioning where required by its certificate.

9.2 The S1–S9 Static Validation Contract

The parameter-selection procedure distinguishes a low-cost proposal stage, *CandidateSuggest*, from *StaticValidate*, which tests a fixed candidate without silently changing its moduli, compression bits, sampler, or error budget. Table 5 summarizes the nine mandatory checks. This is a statement of the reference-model policy, not a claim that all validation software is contained in the present LaTeX source.

A candidate is labeled `STATIC-PASS` only when the specified checks pass under the *same* candidate, policy, and sampler-contract identifiers. A demonstrated failed condition is `STATIC-FAIL`; insufficient evidence is `STATIC-INDETERMINATE`, not a pass. In particular, an S2 table lookup does not imply that the sampler, TILT, or correctness gates have passed. Likewise, an algebraic proof of the protocol does not replace the parameter-dependent reference-sampler certificates.

Structural reuse in S9. The structural gate S9 consists of a common proof check (S9-A) and a candidate-specific check (S9-B). S9-A covers KSK/ModDown algebra, R2M/M2R and selected-lane handling, code-sum cancellation, canonical support/codec rules, and fixed-cohort state transitions. A candidate may reuse this structural result only after matching the exact protocol definition and assumptions of Sections 4 and 5. S9-B then checks the actual limb values, model/auxiliary widths, packing/layout, labels, byte counts, and key-target branches. Neither reuse nor a successful S9-B waives S1–S8. Changes to dropout targets or retry rules lie outside this fixed-cohort S9 certificate and require separate analysis.

Table 5. Required static-validation checks for a fixed-cohort candidate.

Gate	Required evidence or condition
S1	Ring/CRT/NTT admissibility, modulus and bit-width ranges, $N = kB$ in Sparse, and consistent parameter tuple.
S2	Approved classical security-table applicability for R_Q, R_{QP} under the stipulated ssRLWE assumptions, and the exact $QP \le 2^{214}$ check for the cited policy.
S3	Counts for all rounds, cached key material, local rows, admitted outputs, sampler calls, and security-game/held-out samples.
S4	One-lifetime flooding bound, sampler distribution, symmetry, TV discrepancy, and certified second moments, tails, and moment-generating functions.
S5	Uniform PPT conditional-sampler/TILT certificate, finite-precision and rejection budgets, and base/DD ideal-XOF bad-event bounds.
S6	Distributed-dither numerical law and model/auxiliary reference RMS precision.
S7	Final signed-code representability without clipping.
S8	Lifetime numerical tails, modular no-wrap bounds, and total δ_{corr} requirement.
S9	Core algebra, setup/key domains, canonical codec, message serialization, support mapping, and execution-state consistency.

9.3 Whole-Execution Accounting for a Fixed Cohort

The certification counts are determined *before* randomness is drawn; otherwise an error bound could be incorrectly selected after an adaptive support was observed. Let

$$R_m = \lceil d/N \rceil, \qquad R_a = \lceil d_{\mathrm{aux}}/N \rceil, \qquad J = \lceil d/B \rceil, \qquad K_\rho = \max\{1, \lceil \rho d/B \rceil\}.$$

A Full client produces $R_m + R_a$ local ciphertext rows in a round. Sparse $k = 1$ produces at most $K_\rho + R_a$ rows and Sparse $k > 1$ at most $\lceil K_\rho/k \rceil + R_a$ rows. These are the q_x/T quantities used in (91). The same public keys, cached KSK rows and, where needed, M2R shares persist across rounds; *fresh online arithmetic randomness* is still sampled for every invocation. The error ledger therefore counts cache generation once but all online uses and released coefficients over the full lifetime.

For fixed-cohort Sparse, every admitted label has at least τ contributors. Since each of the n clients selects at most K_ρ labels, the total number of admitted labels in any round obeys

$$y_t = |Y_t| \le \min\{J, \lfloor nK_\rho/\tau \rfloor\}. \tag{116}$$

The number of final aggregate groups is therefore at most $\lceil y_t/k \rceil$, but the actual number is determined by the admitted support of that round. For the numerical example below, Full has $R_m = 58$ model blocks and $R_a = 1$ auxiliary block.

Sparse has $J = 231$, $K_\rho = 47$, at most 12 local model packs per client, at most 125 admitted labels, and at most 32 final model groups. These are worst-case resource bounds, not claims that the maximum admission occurs every round.

9.4 Representative Parameters for CIFAR-100

We consider two explicitly specified CIFAR-100 reference configurations, one for Full and one for Sparse, with model precision $p_m = 13$. Their reported static checks were performed for the parameters below. Both configurations use $n = 8$, $c = 1$, $\tau = 3$, $T = 200$, $N = 8192$, $m = 2$, $d = 472756$, $d_{\text{aux}} = 2464$, and

$$\boldsymbol{\alpha} = (13, 15, 17, 16, 14, 19, 18, 16), \qquad \sum_i \alpha_i = 128.$$

The Sparse candidate uses $B = 2048$, $k = 4$, and $\rho = 0.2$. The Full candidate has no sparse support or R2M/M2R. The common base-error width is $\sigma_e = 3.19$. Table 6 records the model/auxiliary configurations used by both candidates.

Explicit moduli and reference implementation. The reported candidate uses

$$\begin{aligned} q_0 &= 281474976694273, \\ q_1 &= 140737488486401, \\ P &= 1152921504606830593, \\ Q = q_0 q_1 &= 39614081291720236145295081473. \end{aligned} \tag{117}$$

The two Q primes are 48-bit and P is 60-bit; all three are $1 \pmod{2N}$, with $\lceil \log_2 Q \rceil = 96$. The exact product has $\log_2(QP) \simeq 155.0000000013 < 214$. The numerical assessment uses an approved classical Category-128 attack-cost table, a finite-rational reference Gaussian sampler, and the canonical codec specified in Subsection 4.8. These choices belong to the displayed parameter tuple; they do not certify arbitrary primes of the same bit lengths.

9.5 Reported Certificate Bounds and Their Interpretation

Table 7 gives the *reported outward-rounded upper bounds* for the two displayed parameter tuples, evaluated with 512-bit Arb intervals. The detailed interval logs and executable validation code are unavailable here; consequently, these values have *not* been independently recomputed. Each required limit is stated separately to keep numerical precision, privacy failure, and correctness failure distinct.

Comparison with the required gates. Both reported flooding bounds are below 2^{-20}. The total noncomputational bounds are below 2^{-18}, with the sampler discrepancy already included in the appropriate lifetime sum rather than added again per draw. Both model RMS bounds are below $2/2^{13}$ and both auxiliary RMS bounds are below $73.02073288/2^{18}$. The model and auxiliary no-clipping

Table 6. Model/auxiliary parameter tuples for the two CIFAR-100 reference candidates.

Parameter	Model	Auxiliary
Dimension	472756	2464
Input bound $U_{\max,\ell}$	2	73.02073288
RMS precision p_ℓ	13	18
Encoding scale Δ_ℓ	$3 \cdot 2^{84}$	2^{80}
(c_1, c_a, c_2, c_d)	$(22, 57, 17, 15)$	$(27, 61, 23, 19)$
Smudging width	2^{78}	2^{72}
Delivery scale $\Delta_{d,\ell}$	8109	3589

Table 7. Reported static-reference upper bounds for the two parameter tuples. Sparse uses $k = 4$, $B = 2048$; no Sparse $k = 1$ certificate is implied.

Quantity	Full-SD	Sparse-SD ($k = 4$)
Lifetime flooding ϵ_F	5.2331159×10^{-7}	4.1816280×10^{-7}
Sampler TV / draw	$3.9216655 \times 10^{-114}$	$3.9216655 \times 10^{-114}$
Noncomputational total	1.0466232×10^{-6}	8.3632561×10^{-7}
TILT trial cap (M/A)	75/91	74/90
Model reference RMS	2.0910428×10^{-4}	2.0910428×10^{-4}
Auxiliary reference RMS	2.5705675×10^{-4}	2.5705675×10^{-4}
Model no-clipping LHS	1.6235517×10^{4}	1.6235395×10^{4}
Auxiliary no-clipping LHS	2.6207798×10^{5}	2.6207798×10^{5}
Lifetime correctness failure	$2.6193448 \times 10^{-10}$	$4.6566129 \times 10^{-10}$
Model no-wrap LHS	1.4870868×10^{28}	1.4870756×10^{28}
Auxiliary no-wrap LHS	1.1299673×10^{28}	1.1299673×10^{28}

limits are, respectively, $2^{15-1} - 1 = 16383$ and $2^{19-1} - 1 = 262143$. Both reported no-wrap left-hand sides are strictly below $Q/2$, and both lifetime correctness-failure bounds are below 2^{-30}. The reported TILT caps are also below the policy limit 128^4.

What the result certifies. Both assessed configurations report `S1-S9=PASS` and `STATIC-PASS`. Because machine-readable validation logs and the executable interval checker are not available with this manuscript, these statuses are *reported static evidence*, not the result of an independent rerun.

The qualification is material: S2 applies an approved estimated attack-cost table and is not an exact numerical upper bound on ϵ_{comp}; S4–S8 concern the declared finite reference-sampler and application-input contracts. No static pass for Sparse $k = 1$ follows from this $k = 4$ case, and neither dropout/retry executions nor an arbitrary deployed binary inherit the certificate. Likewise, the correctness reference RMS bound cannot be extended to unconditional actual-output RMS on exceptional executions without further analysis. These are the limits of the numerical claim, not unreported quantities to be silently set to zero.

Certificate coverage across experimental configurations. All experimental configurations were validated using the S1–S9 static privacy and correctness checks, and only configurations satisfying `STATIC-PASS` were used in our experiments. This validation covers the different chunk sizes, transmission ratios, and client counts considered in Section 10. Table 7 presents detailed certification results for representative Full and Sparse configurations, with $(k, B, \rho) = (4, 2048, 0.2)$ for Sparse. The remaining experimental configurations were validated separately using the same certification procedure.

9.6 Per-Round Online Model Communication

We count logical payload bits across every client uplink and every server-to-client unicast, rather than only the bytes in one representative ciphertext. Setup is accounted for separately, and auxiliary-state traffic is added through its independent dense path. For Sparse, let $g_i = |H_i^{\mathrm{up}}|$ be the number of chunks uploaded by client i, $y_t = |Y_t^\star|$ the number of finally admitted labels, and set

$$R_i = \lceil g_i/k \rceil, \qquad R_t = \lceil y_t/k \rceil.$$

As in the fixed-cohort protocol, every final survivor answers each admitted model-decryption request even if that client did not contribute the corresponding chunk. The resulting model-path payloads per sender or recipient are summarized in Table 8.

The seed of each uploaded local RLWE ciphertext replaces its uniform a-component, whereas only the occupied Sparse b-lanes contribute Bg_i coefficient codes. M2R repacking amortizes a full-polynomial aggregate-a request over up to k admitted labels. Decryption responses and delivered updates carry only the admitted output coefficients, although every participating client sends its own partial response. For one fixed-cohort Sparse round, the model-path total, ignoring small headers and support metadata, is therefore

$$\begin{aligned} \mathrm{Comm}_{m,t}^{\mathrm{SP}} = &\sum_{i \in \mathcal{C}} (128 R_i + B g_i c_1) \\ &+ n R_t N c_a + n y_t B (c_2 + c_d), \end{aligned} \tag{118}$$

when server-to-client distribution is charged as n unicasts. The corresponding Full total is

$$\begin{aligned} \mathrm{Comm}_{m,t}^{\mathrm{F}} = &\, n R_m (128 + N c_1) \\ &+ n R_m N (c_a + c_2) + n d c_d. \end{aligned} \tag{119}$$

For auxiliary updates, use the Full formula with d, R_m replaced by d_{aux}, R_a and the bit widths replaced by the auxiliary tuple. In particular, the auxiliary path is never sparsified merely because the model path uses Sparse.

Support and control metadata. The complete online count adds the per-client shuffled support announcements, admitted-label and upload-support bitmaps, public DD seeds, round identifiers, message headers, and any explicitly charged

Table 8. Per-round online *model-path* logical payload per sender or recipient, in bits; ancillary bitmap/headers are separate.

Message	Sparse	Full
Client upload	$128R_i + Bg_ic_1$	$R_m(128 + Nc_1)$
Decryption request	R_tNc_a	R_mNc_a
Partial response	y_tBc_2	R_mNc_2
Delivered result	y_tBc_d	dc_d

finalization control. An uncompressed length-J bitmap has J logical bits; its replication over multiple clients must not be omitted. The preceding table intentionally describes the dominant model coefficient payload, not all observable wire bytes. This distinction matters most for very small B or small transmission ratios, when metadata and a fixed seed cost become relatively more important. For the fixed-cohort 200-round experiment, cumulative online traffic sums the complete per-round traffic across the entire execution; it is not the communication required to reach the best accuracy at the first occurrence of that accuracy.

9.7 Setup Material and Communication Accounting

Setup communicates the collective public-key shares, KSK rows, and (for Sparse $k > 1$) M2R switching-key shares. Let $L_{QP} = \lceil \log_2(QP) \rceil$ denote the logical coefficient width. One full polynomial of R_{QP} then has a raw logical payload of approximately NL_{QP} bits. Each client sends one public-key share and $2m$ KSK polynomials; each Sparse $k > 1$ client additionally sends km M2R-share polynomials. Consequently a raw uplink estimate for these key contributions is

$$\mathrm{Comm}_{\mathrm{setup,up}} \simeq nNL_{QP}\big(1 + 2m + \mathbf{1}_{\{\mathrm{SP},\,\mathrm{k}>1\}}km\big), \tag{120}$$

subject to the exact canonical serializer and any shared-mask seed reconstruction specified in Section 4. The server also distributes collective public-key material, public CRS seeds, and setup/control messages. A broadcast charged as separate unicasts must be multiplied by the number of recipients. Keys are cached and reused throughout the fixed-cohort execution, so these one-time setup costs must be distinguished from the repeated online traffic.

In Sparse $k > 1$, M2R switching-key shares and their corresponding masks are not retransmitted every round. In contrast, local encryption seeds, compressed uploads, aggregate-a requests, and individual partial-decryption responses are online traffic. The relative setup cost is best reported as the ratio of actual setup bits to cumulative online bits, not inferred from the raw model coefficient formulas alone.

Dropout is a separate operational accounting case. If a client is excluded after upload, the uploaded model ciphertext has already been transmitted; survivor-specific KSK generation or reaggregation may add traffic, while fewer decryption

participants can reduce response traffic. Those events must be included when evaluating the operational dropout path. They are *not* represented by simply substituting $n \mapsto |\mathcal{S}|$ into a fixed-cohort security certificate or by assuming a new runtime cryptographic target is already statically certified.

9.8 Implications for Parameter Selection

Larger c_1, c_a, c_2, c_d increase coefficient payloads, while their reduction may enlarge quantization error and violate statistical security or numerical precision gates. Sparse granularity B has a different trade-off: smaller chunks can select important coordinates more precisely, but increase the number of packed groups, key-switching and M2R operations, and associated processing work. Therefore, communication savings must be evaluated together with label admission, decryption payloads, setup amortization, and the mandatory S1–S9 conditions. The experimental evaluation compares accuracy, communication, and processing costs under the measured training configurations; the present formulas specify what is counted rather than predicting empirical training accuracy.

10 Experimental Evaluation

10.1 Experimental Setup

Implementation and environment. We implemented the Full and Sparse profiles of our threshold-CKKS aggregation protocol using PyTorch and OpenFHE. We simulate the server and clients within a single process, executing client computations sequentially on a machine with an AMD Ryzen Threadripper PRO 5995WX CPU (64 cores) and 512 GB of RAM. The implementation and experimental configurations are publicly available at `https://github.com/pmlc0614/Threshold-CKKS-FL`.

Datasets and training configuration. We evaluate CIFAR-10/ResNet-20 and CIFAR-100/ResNet-32, using plaintext federated learning and FedPHE as baselines. For FedPHE, we follow the experimental configuration provided in its publicly released implementation. Unless otherwise specified, experiments use eight clients and 200 communication rounds. Non-IID partitions are generated from a class-wise Dirichlet distribution with $\alpha = 1.0$. Each client performs one local epoch per round using SGD with momentum 0.9, weight decay 5×10^{-4}, and batch size 128. We apply round-wise cosine learning-rate decay from an initial rate of 0.1 to zero.

Protocol configuration. We use ring dimension $N = 8192$ and a ciphertext modulus comprising two RNS primes. For Sparse, we vary the target transmission ratio ρ and chunk size $B = N/k$, where k is the maximum number of chunks packed into one RLWE ciphertext. Each admitted chunk requires at least $\tau = 3$ contributors. Model and auxiliary updates use separately configured scaling and compression parameters.

Parameter settings. For each experimental configuration, we select scaling, compression, and smudging parameters according to its input bounds and precision requirements, using the representative settings in Table 6 as a reference. The relative RMS precision targets are $p = 13$ for model updates and $p_{\text{aux}} = 18$ for auxiliary updates. Compression bit widths remain near their representative values. For every experimental configuration, we use parameters that pass all S1–S9 static privacy and correctness validation checks (`STATIC-PASS`), including the configurations with different chunk sizes and client counts. Section 9.5 presents detailed certification results for representative Full and Sparse configurations, with $(k, B, \rho) = (4, 2048, 0.2)$ for Sparse. All other experimental configurations were validated using the same certification procedure.

Evaluation metrics. We report the best test accuracy attained during training. Communication is the cumulative online payload over all client uplinks and server-to-client unicasts, including model and auxiliary payloads, seeds, support bitmaps, and decryption and result-delivery messages. Setup traffic is accounted for separately as a percentage of cumulative online communication. We report cumulative online runtime estimated from measured local-training and encrypted-aggregation times under a parallel-client execution model. Setup, evaluation, and network latency are excluded; these estimates are not distributed-system wall-clock measurements.

10.2 Accuracy–Communication Trade-off

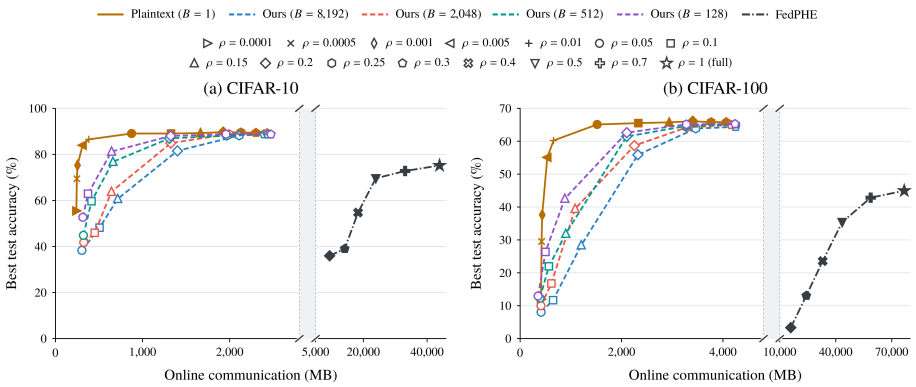


Fig. 2. Best test accuracy versus cumulative online communication for plaintext FL, the proposed Sparse profile, and FedPHE over 200 training rounds. Shaded bands indicate axis breaks.

Fig. 2 compares best test accuracy against cumulative online communication over 200 training rounds. Because the minimum-cohort requirement can admit

different numbers of components even for the same target transmission ratio ρ, we compare methods at matched accuracy levels. The corresponding communication costs are estimated by linear interpolation between adjacent measured points on the accuracy–communication curves. This compares 200-round communication, not the cost of first reaching a target accuracy.

Comparison with FedPHE. At the full-transmission accuracies of FedPHE (75.19% on CIFAR-10 and 44.96% on CIFAR-100), Sparse with $B = 128$ requires approximately 0.55 GB and 0.91 GB, respectively, compared with 43.91 GB and 76.68 GB for FedPHE. This represents approximately 79× and 84× reductions in online communication.

Comparison with plaintext. Full achieves nearly the same accuracy as full-transmission plaintext, with losses of only 0.32 and 0.29 percentage points on CIFAR-10 and CIFAR-100, respectively. Its online communication is only 1.78× and 1.76× that of plaintext, respectively.

For Sparse, we consider target accuracies five percentage points below those of full-transmission plaintext (84.92% on CIFAR-10 and 61.32% on CIFAR-100). At these accuracy levels, Sparse with $B = 128$ requires approximately 0.85 GB and 1.76 GB, respectively, compared with 0.34 GB and 0.85 GB for plaintext. Thus, Sparse incurs only 2.5× and 2.1× the online communication of plaintext, respectively.

Effect of chunk granularity. Smaller chunks enable finer-grained selection of important updates through RLWE–MLWE conversion, improving the accuracy–communication trade-off. At $\rho = 0.2$, reducing B from 8192 to 128 improves accuracy from approximately 81.6% to 88.01% on CIFAR-10 and from 55.9% to 62.61% on CIFAR-100, while slightly reducing communication in both cases. This illustrates the benefit of fine-grained sparsification while retaining compact RLWE packing.

Across the Sparse configurations in Fig. 2, setup communication accounts for less than 10% of cumulative online communication in most cases. The observed range is 0.54–57.30%, with larger ratios at low transmission ratios and small chunk sizes. Complete setup and online uplink/downlink communication and runtime measurements are reported in Tables 11 and 12.

10.3 Online Runtime

Fig. 3 shows that the runtime overhead of encrypted aggregation accounts for less than half of total online runtime in most configurations. Smaller chunks increase this overhead through finer-grained processing but improve accuracy through more selective updates. This highlights the accuracy–runtime trade-off, with $B = 128$ and 512 providing practical balances between accuracy and processing cost in the tested configurations.

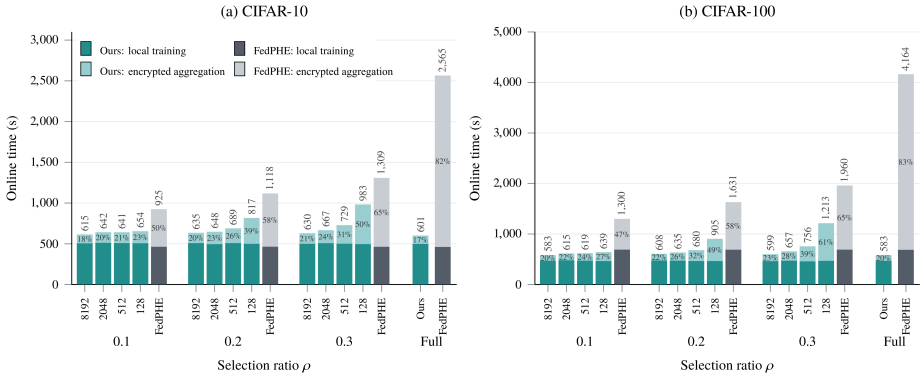


Fig. 3. Estimated cumulative online runtime, broken down into local training and encrypted aggregation, by transmission ratio ρ and chunk size B.

10.4 Client Dropout

We evaluate temporary client dropout on CIFAR-100 using Sparse with $\rho = 0.2$ and $B = 128$. One client is excluded in 5%, 10%, or 20% of the training rounds, corresponding to 10, 20, or 40 randomly selected rounds. Dropout occurs after all clients have uploaded their updates and before the first distributed-decryption request. The client rejoins in the next round unless selected for another dropout event.

Table 9. Effect of client dropout on CIFAR-100.

Dropout (rounds)	Accuracy (%)	Online traffic (MB)	Online time (s)
0% (0)	62.61	2,103.13	904.70
5% (10)	63.41	2,082.86	896.83
10% (20)	61.97	2,080.93	913.11
20% (40)	62.26	2,024.43	896.64

Table 9 shows no substantial accuracy degradation or runtime increase under the evaluated dropout conditions. The largest accuracy loss is 0.64 percentage points, while online runtime varies by less than 1%. Cumulative online communication decreases by 0.96–3.74%: fewer surviving clients reduce collective-decryption traffic, outweighing the additional key-reconfiguration traffic in these experiments.

10.5 Effect of Client Count

Table 10 compares Full and Sparse transmission on CIFAR-100 for $n = 8, 16, 32, 64$. For Sparse, the plaintext and encrypted runs use the same tar-

get ratio $\rho = 0.2$, chunk size $B = 512$, minimum cohort size $\tau = 3$, and quantized aggregation weights. Across these client counts, the maximum accuracy loss relative to plaintext is only 0.42 percentage points for Full and 0.35 percentage points for Sparse.

In Full, per-client online traffic increases by only 7.17%, from 6.709 to 7.190 MB per round. Sparse shows a larger increase, from 1.313 to 5.908 MB, primarily because more selected chunks satisfy the minimum-cohort requirement as n grows. As a result, a larger fraction of selected chunks is admitted for aggregation, effectively increasing the realized transmission ratio despite a fixed target ρ. By contrast, the compression widths c_1, c_2, c_a, c_d each increase by only two bits for both model and auxiliary updates.

Full's online runtime decreases by 59.6% as increasing n reduces each client's local-training workload. Sparse shows a smaller decrease of 3.8%, because the larger encrypted-aggregation workload partially offsets the savings in local training.

Table 10. Effect of client count on CIFAR-100. Accuracy is reported as encrypted (plaintext); traffic and runtime refer to encrypted training.

Mode	n	Accuracy (%)	MB/client/round	Runtime (s)
Full	8	66.03 (66.32)	6.709	583.12
	16	61.11 (60.52)	6.709	354.88
	32	49.48 (49.90)	6.949	266.54
	64	32.47 (32.01)	7.190	235.67
Sparse	8	61.44 (61.18)	1.313	680.47
	16	58.12 (58.18)	3.297	586.73
	32	47.32 (47.41)	5.182	627.29
	64	30.25 (30.60)	5.908	654.58

10.6 Detailed Experimental Results

Tables 11 and 12 provide the complete measured results for the accuracy–communication and runtime comparisons, including one-time setup traffic and the online uplink/downlink breakdown. The data correspond to the experimental configurations summarized above.

Table 11. Detailed accuracy, setup and online communication, and runtime on CIFAR-10.

Method	ρ	B	Acc.	Up (MB)		Down (MB)		Time (s)	
				Setup	Online	Setup	Online	Local	Enc.
Plaintext[a]	**Full**	1	**89.92**	–	1754	–	1754	511	
	0.3	1	89.40	–	642	–	1660	701	
	0.25	1	89.53	–	555	–	1575	689	
	0.2	1	89.68	–	468	–	1458	673	
	0.15	1	89.26	–	381	–	1284	663	
	0.1	1	89.06	–	293	–	1034	627	
	0.05	1	89.07	–	206	–	667	601	
	0.01	1	86.52	–	136	–	246	561	
	0.005	1	83.99	–	128	–	184	544	
	0.001	1	75.26	–	121	–	132	546	
	0.0005	1	69.41	–	120	–	126	551	
	0.0001	1	55.48	–	119	–	120	538	
FedPHE	**Full**	4096	**75.19**	–	30736	–	13173	461	2104
	0.7	4096	72.80	–	21561	–	11539	464	1569
	0.5	4096	69.61	–	15598	–	8310	461	1206
	0.4	4096	54.73	–	12386	–	5954	462	1012
	0.3	4096	38.99	–	9634	–	4516	463	846
	0.2	4096	35.97	–	6423	–	3009	464	654
Ours	**Full**	8192	**89.60**	6.39	2200	10.10	4044	498	103
	0.3	8192	88.79	6.39	711	10.10	1684	499	130
		2048	89.28	16.61	718	10.10	1713	508	160
		512	88.65	47.28	716	10.10	1723	501	228
		128	88.72	169.97	716	10.10	1758	496	487
	0.25	8192	88.19	6.39	621	10.10	1489	506	129
		2048	88.82	16.61	578	10.10	1416	505	158
		512	88.10	47.28	568	10.10	1401	499	208
		128	88.78	169.97	560	10.10	1398	498	417
	0.2	8192	81.55	6.39	413	10.10	988	508	127
		2048	84.91	16.61	383	10.10	942	496	152
		512	86.97	47.28	375	10.10	933	508	181
		128	88.01	169.97	379	10.10	944	500	317
	0.17	128	85.74	169.97	262	10.10	634	494	247
	0.165	128	84.65	169.97	242	10.10	590	491	237
	0.15	8192	60.69	6.39	224	10.10	490	514	118
		2048	63.78	16.61	196	10.10	447	504	140
		512	76.80	47.28	197	10.10	462	509	150
		128	81.25	169.97	191	10.10	452	499	204
	0.1	8192	48.17	6.39	168	10.10	338	506	109
		2048	45.96	16.61	144	10.10	306	512	130
		512	59.65	47.28	131	10.10	283	508	133
		128	62.94	169.97	119	10.10	255	507	147
	0.05	8192	38.32	6.39	113	10.10	190	511	87
		2048	41.78	16.61	110	10.10	215	505	117
		512	44.86	47.28	104	10.10	217	511	124
		128	52.67	169.97	99	10.10	216	491	134

Acc. is best test accuracy (%). Full denotes $\rho = 1$. Setup is one-time communication. Local and Enc. are estimated parallel online times for local training and encrypted-path processing, rounded independently. [a]Plaintext reports total online runtime only.

Table 12. Detailed accuracy, setup and online communication, and runtime on CIFAR-100.

Method	ρ	B	Acc.	Up (MB)		Down (MB)		Time (s)	
				Setup	Online	Setup	Online	Local	Enc.
Plaintext[a]	**Full**	1	**66.32**	–	3041	–	3041	476	
	0.3	1	65.84	–	1113	–	2953	785	
	0.25	1	65.79	–	961	–	2806	769	
	0.2	1	66.08	–	810	–	2593	742	
	0.15	1	65.75	–	659	–	2281	718	
	0.1	1	65.51	–	507	–	1824	669	
	0.05	1	65.10	–	356	–	1163	611	
	0.01	1	60.22	–	235	–	424	540	
	0.005	1	55.09	–	220	–	316	527	
	0.001	1	37.60	–	208	–	227	522	
	0.0005	1	29.53	–	206	–	216	518	
	0.0001	1	11.90	–	205	–	207	519	
FedPHE	**Full**	4096	**44.96**	–	53674	–	23003	687	3476
	0.7	4096	42.84	–	37618	–	21094	691	2551
	0.5	4096	35.40	–	27066	–	16368	688	1896
	0.4	4096	23.56	–	21561	–	11435	688	1575
	0.3	4096	13.04	–	16515	–	7704	690	1270
	0.2	4096	3.33	–	11010	–	4912	689	943
Ours	**Full**	8192	**66.03**	6.39	3789	16.57	6944	468	115
	0.3	8192	64.37	6.39	1259	16.57	2990	462	137
		2048	64.97	16.61	1222	16.57	2944	473	184
		512	65.15	47.28	1236	16.57	3012	458	298
		128	65.27	169.97	1211	16.57	3028	469	744
	0.25	8192	63.86	6.39	997	16.57	2466	471	134
		2048	64.21	16.61	933	16.57	2353	467	180
		512	64.62	47.28	931	16.57	2344	463	261
		128	65.13	169.97	924	16.57	2354	459	608
	0.2	8192	55.88	6.39	665	16.57	1658	477	132
		2048	58.66	16.61	635	16.57	1618	472	163
		512	61.44	47.28	589	16.57	1512	465	215
		128	62.61	169.97	585	16.57	1518	465	440
	0.19	128	62.01	169.97	510	16.57	1310	467	399
	0.185	128	60.75	169.97	477	16.57	1225	473	380
	0.155	128	47.77	169.97	276	16.57	679	463	259
	0.15	8192	28.39	6.39	358	16.57	847	474	127
		2048	39.48	16.61	314	16.57	770	474	153
		512	31.89	47.28	261	16.57	639	465	164
		128	42.55	169.97	255	16.57	626	465	247
	0.1	8192	11.68	6.39	207	16.57	442	468	115
		2048	16.76	16.61	190	16.57	427	479	135
		512	21.97	47.28	172	16.57	396	473	146
		128	26.36	169.97	153	16.57	347	464	175
	0.05	8192	8.02	6.39	143	16.57	270	479	101
		2048	9.99	16.61	133	16.57	278	478	127
		512	12.33	47.28	123	16.57	267	472	133
		128	12.94	169.97	113	16.57	241	473	143

Acc. is best test accuracy (%). Full denotes $\rho = 1$. Setup is one-time communication. Local and Enc. are estimated parallel online times for local training and encrypted-path processing, rounded independently. [a]Plaintext reports total online runtime only.

11 Conclusion

We presented a communication-efficient threshold-CKKS protocol for secure cross-silo federated learning. The framework supports server-side post-encryption weighting and fixed-cohort collusion-aware encrypted aggregation without a trusted key-distribution center. A separately specified operational extension permits dropout-aware reaggregation without retransmitting local ciphertexts; it is not covered by the fixed-cohort whole-view security result. RLWE–MLWE conversion and stage-specific communication compression enable the Sparse profile to aggregate fine-grained model chunks while retaining compact ciphertext packing. Under the explicitly specified leakage and cryptographic assumptions, our analysis establishes whole-view simulation security for fixed-cohort executions without dropout or retry.

Experiments on CIFAR-10 and CIFAR-100 show that Full maintains nearly the same accuracy as plaintext, with losses of only 0.32 and 0.29 percentage points, while requiring only 1.78× and 1.76× the online communication, respectively. At matched accuracy levels, Sparse reduces online communication by approximately 79× and 84× compared with FedPHE. At accuracy targets five percentage points below full-transmission plaintext, Sparse requires only 2.5× and 2.1× the online communication of plaintext, respectively. These results demonstrate the practicality of communication-efficient encrypted aggregation for secure cross-silo FL.

References

1. Bae, Y., Cheon, J.H., Kim, J., Park, J.H., Stehlé, D.: HERMES: Efficient ring packing using MLWE ciphertexts and application to transciphering. In: Advances in Cryptology – CRYPTO 2023. Lecture Notes in Computer Science, vol. 14084, pp. 37–69. Springer (2023). https://doi.org/10.1007/978-3-031-38551-3_2
2. Bell, J., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly)logarithmic overhead. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1253–1269. ACM (2020). https://doi.org/10.1145/3372297.3417885
3. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191. ACM (2017). https://doi.org/10.1145/3133956.3133982
4. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: International conference on the theory and application of cryptology and information security. pp. 409–437. Springer (2017)
5. He, Y., Yu, J.: Toward secure weighted aggregation for privacy-preserving federated learning. IEEE Transactions on Information Forensics and Security **20**, 3475–3488 (2025)
6. Kairouz, P., Liu, Z., Steinke, T.: The distributed discrete gaussian mechanism for federated learning with secure aggregation. In: Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Re-

search, vol. 139, pp. 5201–5212. PMLR (2021), https://proceedings.mlr.press/v139/kairouz21a.html
7. Li, Y., Yan, N., Chen, J., Wang, X., Hong, J., He, K., Wang, W., Li, B.: FedPHE: A secure and efficient federated learning via packed homomorphic encryption. IEEE Transactions on Dependable and Secure Computing **22**(5), 5448–5463 (2025). https://doi.org/10.1109/TDSC.2025.3567301
8. Lu, S., Li, R., Liu, W., Guan, C., Yang, X.: Top-k sparsification with secure aggregation for privacy-preserving federated learning. Computers & Security **124**, 102993 (2023)
9. Ma, J., Naas, S.A., Sigg, S., Lyu, X.: Privacy-preserving federated learning based on multi-key homomorphic encryption. International Journal of Intelligent Systems **37**(9), 5880–5901 (2022)
10. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR (2017), https://proceedings.mlr.press/v54/mcmahan17a.html
11. Mouchet, C., Troncoso-Pastoriza, J., Bossuat, J.P., Hubaux, J.P.: Multiparty homomorphic encryption from ring-learning-with-errors. Proceedings on Privacy Enhancing Technologies **2021**(4), 291–311 (2021)
12. Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Transactions on Information Forensics and Security **13**(5), 1333–1345 (2018). https://doi.org/10.1109/TIFS.2017.2787987
13. So, J., He, C., Yang, C.S., Li, S., Yu, Q., Ali, R.E., Guler, B., Avestimehr, S.: Lightsecagg: A lightweight and versatile design for secure aggregation in federated learning. In: Proceedings of Machine Learning and Systems. vol. 4, pp. 694–720 (2022)
14. Stich, S.U., Cordonnier, J.B., Jaggi, M.: Sparsified SGD with memory. In: Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), https://proceedings.neurips.cc/paper_files/paper/2018/hash/b440509a0106086a67bc2ea9df0a1dab-Abstract.html
15. Wei, X., Huang, R., Lei, L., Wu, J.: FedSHE–CQ: A communication-efficient homomorphic encryption framework for federated learning. Computer Networks **274**, 111813 (2026). https://doi.org/10.1016/j.comnet.2025.111813
16. Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y.: BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning. In: 2020 USENIX Annual Technical Conference (USENIX ATC 20). pp. 493–506. USENIX Association (2020)