SCARFL: SINGLE-CIPHERTEXT PRIVACY-PRESERVING BYZANTINE-ROBUST FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Privacy-preserving Byzantine-robust federated learning requires computing geometric statistics of client updates without exposing the updates themselves. Existing baselines like AegisFL achieve this by extracting encrypted scalars from two coefficient-packed ciphertexts per client, which necessitates an interactive consistency check. In this work, we reveal that the complementary packing is mathematically exactly the inverse-index Galois automorphism of the primary packing. Leveraging this insight, we propose SCARFL, which eliminates the redundant client upload by deriving the second ciphertext view directly at the aggregation server while strictly preserving the encrypted scalar interface. To demonstrate the practical viability of this retained interface, we instantiate it with GHARC, an aggregation rule that discovers update clusters and accepts them only when historical-reference evidence passes a security guard. For 30 clients, our Single-CT protocol halves recurring client uploads and reduces total round traffic by 43.0% on HAR and 45.7% on MNIST. Mean round time falls by 11.0% and 9.2%, respectively, with a negligible maximum inner-product error of 2.17×10^{-9}. Under evaluated untargeted, label-flipping, and scaling attacks, the GHARC instantiation maintains stable clean accuracy and suppresses attack success rates with up to 14 malicious clients. Comprehensive ablations validate the roles of its clustering, clipping, ranking, and adaptive core expansion mechanisms.

1 INTRODUCTION

Federated learning lets clients jointly train a model without sharing their local datasets (McMahan et al., 2017). The aggregation server receives updates that can reveal private data (Geiping et al., 2020; Zhu et al., 2019), while Byzantine clients can poison the learned model (Fang et al., 2020; Bagdasaryan et al., 2020). Encrypted robust aggregation addresses both problems but must recover enough information to distinguish useful updates from malicious ones. AegisFL (Chen et al., 2024) exposes encrypted norms, inner products, and means through coefficient-packed CKKS ciphertexts. Each client sends two packings of its update, and a consistency exchange checks that the pair represents the same vector.

SCARFL removes this duplication. In the cyclotomic ring $\mathbb{Z}[X]/(X^N + 1)$, the two AegisFL coefficient packings are related by the automorphism $X \mapsto X^{-1}$. The aggregation server derives the second ciphertext from the first using a public Galois key. The client sends one encrypted update; the scalar-statistic and masked key-conversion interfaces remain available. Because the derived view is tied to the uploaded view, the independent two-view inconsistency addressed by AegisFL's exchange is eliminated. Server-side derivation adds key-switching work and noise, included in the measurements.

GHARC is a robust aggregation rule built from authorized norm and inner-product outputs. It uses historical-reference similarity to select a small core, runs HDBSCAN on each client's similarities to that core, and accepts a candidate according to clipped reference proximity, group size, and core alignment. A guard triggers core expansion when the current evidence is insufficient. The two stages separate discovery of geometrically coherent clients from the decision to trust them.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

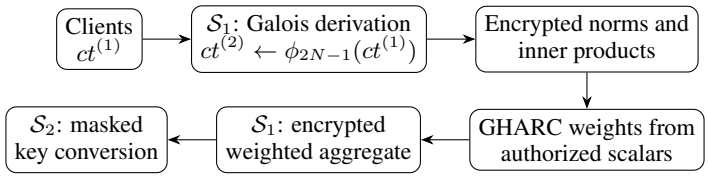

Figure 1: Single-CT processing path. Clients send one ciphertext view; $\mathcal{S}_1$ derives the complementary view before the inherited encrypted scalar interface. GHARC consumes prescribed scalar outputs to weight encrypted updates.

SCARFL provides: (i) an exact packing identity and a Single-CT realization with a narrow, explicit privacy boundary; (ii) GHARC, whose decisions require only the retained encrypted scalar interface; and (iii) matched communication, numerical-error, runtime, robustness, and ablation measurements. The principal comparison keeps CKKS parameters and downstream scalar operations fixed while changing how the second ciphertext view is obtained. We evaluate 30-client IID MNIST and subject-partitioned HAR, and use the three attack categories of AegisFL: untargeted random updates, label flipping, and a triggered scaling attack. Complete protocols, parameter details, and full sweeps appear in the appendix.

2 Setting and Single-Ciphertext Protocol

We use the AegisFL two-server setting. Clients encrypt updates for aggregation server $\mathcal{S}_1$; non-colluding key-holder $\mathcal{S}_2$ helps release prescribed scalar statistics and convert the encrypted aggregate back to the client-key domain. Both servers are honest-but-curious. Up to f clients may submit arbitrary updates. Released norms, inner products, and the final model remain part of the inherited leakage interface; the security statement concerns only the additional view derivation.

For a vector $v = (v_0, \ldots, v_{N-1})$, define the *symbolic* coefficient packings in $A[X]/(X^N + 1)$ by

$$p_1(v) = \sum_{i=0}^{N-1} v_i X^i, \qquad p_2(v) = -\sum_{i=0}^{N-1} v_i X^{N-i}. \tag{1}$$

Since $X^{-i} = -X^{N-i}$ for $1 \le i < N$ and $-X^N = 1$, their exact relation is

$$p_2(v) = p_1(v)(X^{-1}) = \phi_{2N-1}(p_1(v)) \pmod{X^N + 1}. \tag{2}$$

Here $2N - 1 \equiv -1 \pmod{2N}$ is the nonnegative Galois index. The identity is exact at the symbolic layer. With the same sign-symmetric coefficientwise rounding for both packings, it also holds for the scaled encoded plaintexts. At the ciphertext level, CKKS evaluation remains approximate because automorphism and key switching add noise (Appendix G).

In Dual-CT, a client encrypts and uploads both packed views. In Single-CT, it uploads only $ct^{(1)} = \mathrm{Enc}(p_1(v))$. Server $\mathcal{S}_1$ computes

$$ct^{(2)} = \mathrm{KeySwitch}\Big(\phi_{2N-1}(ct^{(1)}), GK_{2N-1}\Big) \tag{3}$$

and supplies this view to the same encrypted norm, inner-product, and mean primitives. In particular, the constant coefficient of a product of the first packing of a and the second packing of b encodes $\langle a, b\rangle$ up to CKKS error. The server also removes the Dual-CT consistency exchange: Eq. (3) fixes the second view for each uploaded first view. A poisoned first view still requires robust aggregation.

Security boundary. Let V be $\mathcal{S}_1$'s first-view ciphertexts, public keys, and local randomness, and let $D(V)$ be the locally derived second views. For any distinguisher using $(V, D(V))$, another distinguisher with V alone can compute $D(V)$ and run the same test. Thus local derivation creates no independent plaintext channel. The prescribed scalar results and final model remain part of the inherited leakage profile; $\mathcal{S}_2$ receives no additional derivation message. Table 1 makes the role-specific boundary explicit. The argument is relative to the inherited interface: it neither supplies a simulation proof for the full interactive scalar and key-conversion interface nor bounds cumulative

leakage from released scores. Appendix H gives the formal scope. Collusion between $\mathcal{S}_1$ and $\mathcal{S}_2$ violates the inherited two-server privacy assumption.

Table 1: Server views under the inherited non-collusion assumption. "Scalar results" are authorized norms, inner products, or means; the final client-key ciphertext is available to $\mathcal{S}_1$.

Role	Observes	Does not receive in plaintext
$\mathcal{S}_1$	Client ciphertexts, local Galois outputs, scalar results, key-conversion mask	Individual updates or the unmasked aggregate
$\mathcal{S}_2$	Compressed scalar-decryption objects, scalar results, masked aggregate during key conversion	Client-uploaded ciphertexts or the unmasked aggregate

The key-conversion mask is sampled by $\mathcal{S}_1$ over the finite plaintext ring at the current CKKS level. Server $\mathcal{S}_2$ decrypts the masked aggregate and returns a client-key encryption; $\mathcal{S}_1$ removes the mask homomorphically. The final model is intentionally decryptable by clients. These inherited interfaces determine what the relative privacy statement can claim; the local Galois step neither changes their authorized scalar outputs nor supplies $\mathcal{S}_2$ with an extra view of client updates.

3 GHARC: Reference-Guided Robust Aggregation

At round t, let g_i^t be client i's update and r^t the most recent nonzero accepted aggregate. At the first round, before such an aggregate exists, GHARC uses a seed-controlled random core; there is no separate clean initialization round. When a guard returns a zero update, the previous nonzero reference is retained. Otherwise, the a clients with the largest $\cos(g_i^t, r^t)$ form a reference core A^t. GHARC represents every client by its similarities to that core,

$$h_i^t = \left[\cos(g_i^t, g_j^t)\right]_{j \in A^t}, \tag{4}$$

and applies HDBSCAN to these vectors. Each eligible nonoutlier cluster is a candidate. The full vector preserves the pattern of agreement with individual core members; its median s_i^t summarizes alignment for candidate ranking. This geometry uses only released inner products and norms, not plaintext update coordinates.

Coherence alone does not justify acceptance: a group of coordinated malicious clients can also form a dense cluster. GHARC measures each client's distance from the historical reference, $d_i^t = \|g_i^t - r^t\|_2$, and sets $S_t = \operatorname{median}_i d_i^t$. For a candidate c, its deliberately *unnormalized* clipped weight and retained mass are

$$\omega_{i,c}^t = \frac{\mathbf{1}[i \in c]}{|c|} \min\left(1, \frac{S_t}{d_i^t}\right), \qquad W_c^t = \sum_i \omega_{i,c}^t. \tag{5}$$

The implementation handles zero distance by the natural clipping limit. Lower W_c^t means less of the candidate's mass survives reference-based clipping. Eligible candidates are ranked by

$$R_c^t = 0.75 W_c^t + 0.15|c|/U + 0.10\widetilde{s}_c^t + b_c, \tag{6}$$

where $\widetilde{s}_c^t$ is the min–max normalized median of s_i^t within c, and $b_c = 0.02$ only for HDBSCAN's initially preferred cluster. A candidate must pass $W_c^t \geq 0.2$; the highest-ranked passing candidate defines $\bar{g}^t = \sum_i \omega_{i,c}^t g_i^t$. This guard ties acceptance to the historical reference, while the weights retain the clipping effect on aggregate magnitude.

The initial core has $a = 7$ clients. If no candidate passes, GHARC expands it by five clients, recomputes similarities and candidates, and repeats until acceptance or $a = U$. If all attempts fail, the update is zero and the previous reference remains in force. This makes expansion conditional on failed evidence rather than a fixed larger-core cost. HDBSCAN settings, deterministic fallbacks, tie handling, and a full round algorithm are in Appendix B.4. Server $\mathcal{S}_1$ forms plaintext weights from authorized scalar outputs, then applies them to encrypted updates by ciphertext-plaintext multiplication; it does not decrypt individual updates.

Algorithm 1 GHARC decision path for one round

Input: encrypted client updates; released norms and inner products; last accepted reference r^t
$a \leftarrow 7$; choose a seeded core if no accepted reference exists
while $a \leq U$ **do**
  select an a-client core by reference cosine; form features h_i^t by Eq. (4)
  discover eligible HDBSCAN clusters c; compute W_c^t and R_c^t by Eqs. (5)–(6)
  if some c has $W_c^t \geq 0.2$ **then**
    return the encrypted weighted aggregate for the highest-ranked passing c
  end if
  if $a = U$ **then**
    return a zero update and retain the prior accepted reference
  end if
  $a \leftarrow \min(a + 5, U)$
end while

Two-stage decision rule. HDBSCAN groups clients by the *shape* of their similarities to the core. It does not determine whether a dense group is benign: attackers submitting mutually aligned updates may form a convincing cluster. The clipped mass W_c^t supplies a separate test against the last accepted direction and scale, while the rank in Eq. (6) resolves competition among candidates that pass the guard. The $|c|/U$ term rewards support but cannot alone override low retained mass, and the core-alignment term favors a group that agrees with the selected reference clients. Because the weights are not renormalized, a borderline accepted candidate also contributes a smaller model update. The ablation in Table 11 separates these mechanisms; high clean accuracy alone can conceal acceptance of the scaling attack.

The reference is updated only after a nonzero accepted aggregate, avoiding the immediate use of a guarded zero vector as the next reference. At the first round, the seed-controlled core is a bootstrap device within the attack-aware round, not a separate benign training round. When HDBSCAN produces only outliers, the implementation applies a deterministic anchor-score fallback; the full threshold and tie rules are documented in Appendix B.4. The fallback preserves deterministic behavior without changing the scalar interface: all scores still come from encrypted norms and inner products. These choices define an empirical defense for the tested attacks; arbitrary coordinated or adaptive Byzantine updates remain outside the evaluation.

Encrypted-statistic workload. GHARC's core representation avoids forming the full pairwise geometry in the common path. With $U = 30$ and initial core size $a = 7$, one round requests 243 scalar results: one reference norm, 30 client norms, 30 client-to-reference products, and 182 distinct client-to-core products after self-pairs and symmetric core–core duplicates are removed. The leading client-to-core term scales as Ua for a fixed core size, whereas a full pairwise Gram matrix has quadratic client count. This explains why the aggregation rule matters to the protocol-level savings: GHARC's measured statistics channel is 81.24 MB per round, between the smaller ShieldFL (Ma et al., 2022) workload and the 160.45 MB pairwise workload of M-FLAME (Chen et al., 2024). The 243-result figure describes the initial-core path; rejected candidates trigger extra encrypted similarities when GHARC expands its core. The complete communication accounting in Appendix E.2 reports this expansion cost instead of using Ua as a worst-case bound.

4 EVALUATION

Protocol and attack setup. The prototype uses a modified Microsoft SEAL coefficient-packing backend with $N = 8192$, CKKS scale 2^{40}, and a 200-bit modulus chain. Measurements run on a 6-core Intel i5-11400 CPU with 32 GB RAM. We use 30 clients, 2,000 measured rounds, 10 local iterations, local learning rate 0.03, and server learning rate 0.5. MNIST is IID; HAR is partitioned by subject. Primary robustness comparisons use seed 42; the seed-43 plaintext GHARC sweep also appears in the main text, and Appendix I reports the independent seed-41 and seed-43 sweeps. Malicious-client counts are $f \in \{3, 9, 12, 14\}$. Following AegisFL (Chen et al., 2024), we evaluate untargeted (UT) Gaussian updates, label flipping $\ell \mapsto M - 1 - \ell$, and scaling attack. For scaling, malicious clients add triggered copies of 20% of local samples, set every 20th feature to zero and

target label to 2, then multiply submitted updates by 5. Scaling results report clean accuracy (Acc) and triggered-sample attack success rate (ASR). The tables use means over rounds 1990–1999. Full settings and all robustness cells are in Appendix E.4.

Efficiency and numerical fidelity. Single-CT replaces one client-supplied ciphertext and two compressed consistency objects per encrypted chunk with server-side derivation. If an update occupies n_c chunks, the U-client recurring saving is approximately $Un_c(B_{\mathrm{ct}} + 2B_{\mathrm{comp}})$, where B_{ct} is a full ciphertext's serialized size and B_{comp} is one compressed consistency object's size. Table 2 gives the measured GHARC totals. Client ciphertext upload is halved; the *whole-round* reduction is 43.0% on HAR and 45.7% on MNIST because unchanged scalar statistics, key conversion, and model delivery also contribute traffic. Absolute savings are 180.52 and 260.75 MB per round. These are serialized protocol bytes, excluding network latency and one-time Galois-key distribution. End-to-end time includes Galois work and falls by 11.0% and 9.2%, respectively. Under matched CKKS parameters, Single-CT's maximum measured encrypted inner-product error is 2.17×10^{-9} across both datasets; the Dual-CT maximum is 5.03×10^{-10}. The maximum drift of the client-decrypted aggregate from its plaintext weighted sum is 2.95×10^{-9}. Full channel, error, and stage breakdowns appear in Appendices E.2, E.3, and K.

Table 2: GHARC protocol cost at 30 clients. Upload is client ciphertext uplink per client (MB); traffic is total serialized protocol MB per round; time is mean seconds per round. Savings compare Single-CT with Dual-CT.

Dataset	Protocol	Upload	Traffic	Time	Time saving
HAR	Dual-CT	6.02	419.79	9.120	–
	Single-CT	3.01	239.27	8.121	11.0%
MNIST	Dual-CT	8.69	570.26	15.271	–
	Single-CT	4.35	309.51	13.872	9.2%

Where the bytes and seconds go. The total in Table 2 includes U times client upload, consistency relay, and model downlink, plus system-wide scalar-statistic and key-conversion messages. Table 3 exposes the channels that change. The removed consistency traffic consists of two compressed objects per update chunk; in this serialization their combined size is approximately one full ciphertext, but they are separate objects. Scalar-statistic traffic remains 81.24 MB per round for the measured GHARC initial-core path on both datasets and grows when a round expands the core. The downlink and masked key-conversion channels are shared by both protocols.

Table 3: GHARC recurring communication breakdown (MB). Upload, consistency, and downlink are per client; scalar statistics and key conversion are system-wide per round. There are $U = 30$ clients.

Dataset	Protocol	Upload	Consistency	Downlink	Statistics	Key conv.
HAR	Dual-CT	6.02	3.01	2.12	81.24	4.24
	Single-CT	3.01	0.00	2.12	81.24	4.24
MNIST	Dual-CT	8.69	4.35	3.06	81.24	6.12
	Single-CT	4.35	0.00	3.06	81.24	6.12

The runtime breakdown in Table 4 explains why halving client ciphertext upload yields a smaller end-to-end speedup. On HAR, Single-CT saves 1.074 seconds of upload and 0.437 seconds of consistency work but spends 0.519 seconds on Galois derivation; on MNIST the corresponding values are 1.562, 0.618, and 0.763 seconds. Encrypted scalar statistics dominate both paths and are nearly unchanged. These measurements average the 12 runs per dataset formed by three attacks and $f \in \{3, 9, 12, 14\}$; they exclude setup and network latency.

Table 4: Selected GHARC runtime stages (seconds per round, 12-run means). The total also includes local training, encrypted aggregation, decryption/update, and evaluation.

Dataset	Protocol	Upload	Consistency	Galois	HE statistics	Total
HAR	Dual-CT	2.152	0.437	0	6.041	9.120
	Single-CT	1.078	0	0.519	6.035	8.121
MNIST	Dual-CT	3.138	0.618	0	8.747	15.271
	Single-CT	1.576	0	0.763	8.770	13.872

Runtime across aggregation rules. Table 5 broadens the comparison beyond GHARC. The same Single-CT substitution reduces mean round time under each evaluated AGR, but the percentage varies with the unchanged encrypted-statistic workload. FedAvg spends relatively little time on robust statistics and shows the largest percentage reduction; M-FLAME's pairwise work dominates its total and shows the smallest. GHARC lies between them. These are CPU wall-clock measurements under the reported process configuration, not network-speed predictions.

Table 5: Mean per-round time across aggregation rules (seconds). Each row averages three attacks and $f \in \{3, 9, 12, 14\}$; saving is relative to Dual-CT.

Dataset	AGR	Single-CT	Dual-CT	Saving
HAR	FedAvg	1.657	3.185	48.0%
	ShieldFL	4.438	5.429	18.2%
	M-FLAME	14.487	15.605	7.2%
	GHARC	8.121	9.120	11.0%
MNIST	FedAvg	4.372	6.535	33.1%
	ShieldFL	8.337	9.744	14.4%
	M-FLAME	22.685	24.107	5.9%
	GHARC	13.872	15.271	9.2%

The inner-product check samples 30 model-gradient vectors per dataset, normalizes them to norm 5, and evaluates self and cross pairs over three seeds. Table 6 shows that the extra Galois step increases absolute error but keeps it many orders of magnitude below the 10^{-3} correctness threshold. The paired protocol comparison uses the same downstream encrypted operators; the difference isolates view construction.

Table 6: Encrypted inner-product error across three seeds. Each self row aggregates 90 pairs and each cross row 150 pairs; maxima are absolute error.

Dataset	Pair	Dual-CT max.	Single-CT max.
HAR	Self	3.98×10^{-10}	2.17×10^{-9}
	Cross	3.05×10^{-10}	1.74×10^{-9}
MNIST	Self	3.28×10^{-10}	2.10×10^{-9}
	Cross	5.03×10^{-10}	2.08×10^{-9}

Robustness at $f = 14/30$. Table 7 compares Single-CT aggregation rules at $f = 14/30$. GHARC has scaling-attack ASR of 0.05% on HAR and 0.48% on MNIST while maintaining 95.70% and 95.33% clean accuracy. FedAvg and ShieldFL retain high clean accuracy in several scaling cells but also have ASR above 99% on HAR and at least 99.17% on MNIST. M-FLAME suppresses scaling on MNIST but not HAR at this count. GHARC's label-flipping accuracy exceeds the other encrypted AGRs on HAR; ShieldFL has higher MNIST UT accuracy. Across all $f \in \{3, 9, 12, 14\}$ GHARC configurations, the maximum scaling ASR is 0.14% on HAR and 0.60% on MNIST in the seed-42 sweep. The seed-41 and seed-43 plaintext sweeps give maxima of $0.40/0.55\%$ and $0.18/0.73\%$ on HAR/MNIST, respectively. The complete Plain/Single-CT table and both additional seeds are in Appendices E.4 and I; the three GHARC plaintext seeds characterize the tested settings rather than establish general robustness.

Table 7: Seed-42 Single-CT robustness at $f = 14/30$ (last-10-round means, percent). Scaling reports clean Acc/triggered-sample ASR; label flipping and UT report clean Acc.

Dataset	AGR	Scaling Acc/ASR	Flipping Acc	UT Acc
HAR	FedAvg	97.78/99.67	57.97	86.64
	M-FLAME	96.27/99.62	95.63	87.05
	ShieldFL	95.88/99.59	36.38	92.17
	GHARC	95.70/0.05	95.56	95.52
MNIST	FedAvg	97.12/99.17	68.57	82.89
	M-FLAME	95.45/0.49	95.45	91.50
	ShieldFL	97.73/99.39	47.46	97.07
	GHARC	95.33/0.48	95.35	95.02

The $f = 14$ slice omits the dependence on malicious fraction. Table 8 retains the complete seed-42 Single-CT GHARC sweep in the main text. HAR scaling ASR stays at or below 0.14% and MNIST scaling ASR stays at or below 0.54% in these encrypted runs; clean accuracy changes modestly across f. The larger 0.60% seed-42 MNIST bound above includes the plaintext row. The evidence covers the stated attacks and client partitions; it does not define a universal Byzantine threshold.

Table 8: Seed-42 Single-CT GHARC sweep (last-10-round means, percent). Scaling cells are clean Acc/triggered-sample ASR; flipping and UT cells are clean Acc.

Dataset	Attack	$f = 3$	$f = 9$	$f = 12$	$f = 14$
HAR	Scaling	96.08/0.14	95.91/0.04	95.57/0.05	95.70/0.05
	Flipping	96.42	95.89	95.81	95.56
	UT	95.95	95.79	95.61	95.52
MNIST	Scaling	95.65/0.51	95.32/0.54	95.21/0.54	95.33/0.48
	Flipping	95.60	95.42	95.43	95.35
	UT	95.55	95.40	95.29	95.02

Robustness under view derivation. Table 9 compares Single-CT and Dual-CT under the same seed, attack, client count, and round budget for all $f \in \{3, 9, 12, 14\}$ runs. Scaling Acc/ASR and the clean accuracies under flipping and UT closely track across protocols. The largest absolute gap across these 24 paired cells is below 0.34 percentage point in clean accuracy and 0.12 percentage point in scaling ASR. At $f = 14$, Dual-CT GHARC reaches $95.57/0.05\%$ scaling Acc/ASR on HAR and $95.49/0.59\%$ on MNIST. The protocols remain numerically distinct because Single-CT adds the measurable Galois noise in Table 6. At this resolution, the observed robustness outcome is preserved under the protocol substitution.

Table 9: Seed-42 GHARC Single-CT/Dual-CT comparison (last-10-round means, percent). Scaling is clean Acc/triggered-sample ASR; flipping and UT are clean Acc.

Dataset	Attack	Protocol	$f = 3$	$f = 9$	$f = 12$	$f = 14$
HAR	Scaling	Single-CT	96.08/0.14	95.91/0.04	95.57/0.05	95.70/0.05
		Dual-CT	96.26/0.18	95.86/0.08	95.74/0.07	95.57/0.05
	Flipping	Single-CT	96.42	95.89	95.81	95.56
		Dual-CT	96.37	96.10	95.64	95.39
	UT	Single-CT	95.95	95.79	95.61	95.52
		Dual-CT	96.28	95.61	95.71	95.50
MNIST	Scaling	Single-CT	95.65/0.51	95.32/0.54	95.21/0.54	95.33/0.48
		Dual-CT	95.65/0.51	95.32/0.54	95.25/0.55	95.49/0.59
	Flipping	Single-CT	95.60	95.42	95.43	95.35
		Dual-CT	95.61	95.43	95.46	95.45
	UT	Single-CT	95.55	95.40	95.29	95.02
		Dual-CT	95.55	95.35	95.33	95.22

Additional plaintext seeds. Independent seed-41 and seed-43 GHARC Plain sweeps use the same datasets, partition types, attack definitions, f values, and 2,000-round budget as seed 42. Table 10 retains the full seed-43 sweep in the main text; Appendix I gives both additional sweeps. Seed 41 reaches at most 0.40% scaling ASR on HAR and 0.55% on MNIST; the seed-43 maxima are 0.18% and 0.73%. At $f = 14$, seed-41 clean accuracy ranges from 94.98% to 95.35% across the six dataset/attack cells. These extra seeds evaluate GHARC *in plaintext*; they do not replicate the competing aggregation rules or the encrypted protocol. They assess consistency under the tested settings and do not support cross-method superiority claims.

Table 10: Seed-43 plaintext GHARC sweep (last-10-round means, percent). Scaling is clean Acc/triggered-sample ASR; flipping and UT are clean Acc.

Dataset	Attack	$f = 3$	$f = 9$	$f = 12$	$f = 14$
HAR	Scaling	96.58/0.18	95.91/0.17	95.95/0.10	95.36/0.04
	Flipping	96.93	95.94	96.10	95.33
	UT	96.77	96.28	95.90	95.19
MNIST	Scaling	95.51/0.60	95.47/0.62	95.48/0.56	95.46/0.73
	Flipping	95.54	95.49	95.45	95.25
	UT	95.38	95.28	95.32	95.01

Mechanism ablation. The seed-42 plaintext $f = 14$ ablation isolates one GHARC component at a time (Table 11). Without HDBSCAN or clipping, scaling ASR rises above 98% on both datasets despite high clean accuracy. Omitting cluster ranking raises HAR ASR from 0.05% to 99.62%. Omitting adaptive expansion leaves MNIST with 1,788 guarded zero-update rounds and lowers clean accuracy to 90.49%. Full GHARC expands only 32 times on MNIST and never on HAR. Weight renormalization changes the final means little. The full-GHARC ablation trajectory reuses the seed-42 plaintext baseline from the robustness sweep.

Table 11: Seed-42 plaintext GHARC ablation under scaling attack at $f = 14/30$. Acc/ASR are last-10-round percentages; Zero counts guarded zero-update rounds among 2,000 rounds. Complete expansion counts are in Appendix E.5.

	HAR			MNIST		
Variant	Acc	ASR	Zero	Acc	ASR	Zero
Full GHARC	95.55	0.05	0	95.49	0.60	0
Without HDBSCAN	96.53	99.62	0	95.52	98.58	0
Without clipping	95.37	99.67	0	96.54	99.28	0
Without ranking	96.64	99.62	0	95.47	0.65	0
Without expansion	95.55	0.05	0	90.49	0.98	1788
Normalized weights	95.55	0.05	0	95.48	0.58	0

5 Related Work and Limitations

AegisFL (Chen et al., 2024) is the closest protocol baseline because its encrypted scalar and masked key-conversion interfaces are retained here. Its two client-supplied coefficient views make encrypted similarity calculations possible but require a separate cross-view consistency check. SCARFL changes the view-construction step while preserving the resulting norm, inner-product, and mean interface. This is why a Dual-CT comparison with the same aggregation rule is more informative about the protocol substitution than a comparison between different AGRs alone.

Related privacy-preserving Byzantine-robust FL systems pursue different points in the design space. Lancelot (Jiang et al., 2025) uses encrypted sorting with a richer homomorphic computation path; DBFL (Chen et al., 2025), BPFLH (Zhu et al., 2026), and PEAR (Sun et al., 2025) develop alternative encrypted distance, density, or trust statistics. Functional-encryption systems such as SAMFL (Guan et al., 2024) and BSR-FL (Zeng et al., 2024) change the underlying cryptographic interface. GHARC addresses a different constraint: it constructs candidate and acceptance decisions from the scalar

outputs already exposed by AegisFL-style coefficient packing. This interface constraint shapes both its core-similarity representation and the reference-distance clipping rule. Appendix F compares the mechanisms in more detail.

The evaluation covers two lightweight datasets, one seed per primary comparison cell, two additional plaintext GHARC seeds, and the three stated attacks. Adaptive attacks against core selection or guard thresholds, larger models, network latency, and cumulative information leakage from repeated scalar releases remain untested. The relative privacy argument leaves compositional security of the inherited interactive primitives unresolved. These conditions bound the empirical and security claims.

6 Conclusion

The inverse-index packing identity lets an aggregation server reconstruct AegisFL's complementary ciphertext from one client upload while preserving the scalar-statistic interface. GHARC uses that interface for reference-guided aggregation. The measurements show lower recurring communication and runtime with small numerical error; the robustness and ablation results characterize the evaluated instantiation.

Reproducibility statement. Appendix E.1 records parameters and measurement procedures; Appendices E.4, E.5, I, and J provide complete reported robustness results. The implementation uses the stated coefficient-packing backend and seeds.

References

Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. In *Protecting privacy through homomorphic encryption*, pp. 31–62. Springer, 2022.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*, pp. 2938–2948. PMLR, 2020.

Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Dong Chen, Hongyuan Qu, and Guangwu Xu. Aegisfl: Efficient and flexible privacy-preserving byzantine-robust cross-silo federated learning. In *Forty-first International Conference on Machine Learning*, 2024.

Xiaoli Chen, Youliang Tian, Shuai Wang, Kedi Yang, Wei Zhao, and Jinbo Xiong. Dbfl: Dynamic byzantine-robust privacy preserving federated learning in heterogeneous data scenario. *Information Sciences*, 700:121849, 2025.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 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.

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to Byzantine-Robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1605–1622. USENIX Association, August 2020. ISBN 978-1-939133-17-5. URL `https://www.usenix.org/conference/usenixsecurity20/presentation/fang`.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

Ke Gu, Wenwu Zhao, Jingjing Tan, Zheng Yang, Xiong Li, and WeiJia Jia. Two-dimensional privacy-preserving federated learning scheme against poisoning attacks. *IEEE Transactions on Dependable and Secure Computing*, 2026.

Menghong Guan, Haiyong Bao, Zhiqiang Li, Hao Pan, Cheng Huang, and Hong-Ning Dai. Samfl: Secure aggregation mechanism for federated learning with byzantine-robustness by functional encryption. *Journal of Systems Architecture*, 157:103304, 2024.

Siyang Jiang, Hao Yang, Qipeng Xie, Chuan Ma, Sen Wang, Zhe Liu, Tao Xiang, and Guoliang Xing. Towards compute-efficient byzantine-robust federated learning with fully homomorphic encryption. *Nature Machine Intelligence*, 7(10):1657–1668, 2025.

Xiaoyuan Liu, Hongwei Li, Guowen Xu, Zongqi Chen, Xiaoming Huang, and Rongxing Lu. Privacy-enhanced federated learning against poisoning adversaries. *IEEE Transactions on Information Forensics and Security*, 16:4574–4588, 2021.

Zhuoran Ma, Jianfeng Ma, Yinbin Miao, Yingjiu Li, and Robert H Deng. Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning. *IEEE Transactions on Information Forensics and Security*, 17:1639–1654, 2022.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. Pmlr, 2017.

Yinbin Miao, Ziteng Liu, Hongwei Li, Kim-Kwang Raymond Choo, and Robert H Deng. Privacy-preserving byzantine-robust federated learning via blockchain systems. *IEEE Transactions on Information Forensics and Security*, 17:2848–2861, 2022.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 739–753. IEEE, 2019. doi: 10.1109/SP.2019.00065.

Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. FLAME: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 1415–1432, Boston, MA, August 2022. USENIX Association. ISBN 978-1-939133-31-1. URL https://www.usenix.org/conference/usenixsecurity22/presentation/nguyen.

Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Ndss*, 2021.

Han Sun, Yan Zhang, Huiping Zhuang, Jiatong Li, Zhen Xu, and Liji Wu. Pear: privacy-preserving and effective aggregation for byzantine-robust federated learning in real-world scenarios. *The Computer Journal*, 68(9):1087–1104, 2025.

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in neural information processing systems*, 33:16070–16084, 2020.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgyS0VFvr.

Kaiping Xue, Jiachen Li, Rui Xue, Yingjie Xue, and Jingcheng Zhao. RobustPPFL: A secure and robust privacy-preserving federated learning framework against poisoning attacks. *IEEE Transactions on Network Science and Engineering*, 13:3351–3368, 2026. doi: 10.1109/TNSE.2025.3632902.

Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Practical packing method in somewhat homomorphic encryption. In *Data Privacy Management and Autonomous Spontaneous Security*, pp. 34–50. Springer, 2014. doi: 10.1007/978-3-642-54568-9_3.

Honghong Zeng, Jie Li, Jiong Lou, Shijing Yuan, Chentao Wu, Wei Zhao, Sijin Wu, and Zhiwen Wang. BSR-FL: An efficient byzantine-robust privacy-preserving federated learning framework. *IEEE Transactions on Computers*, 73(8):2096–2110, 2024. doi: 10.1109/TC.2024.3404102.

Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of dataset properties in Multi-Party machine learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2687–2704. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/zhang-wanrong.

Guofu Zhu, Wenting Shen, Zhiquan Liu, Jing Qin, and Jixin Ma. Bpflh: Byzantine-robust privacy-preserving federated learning for heterogeneous data. *IEEE Transactions on Dependable and Secure Computing*, 2026.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

LLM USAGE DISCLOSURE

Large language models assisted with manuscript drafting and code development. The authors are responsible for the final text, implementation, results, and claims.

A THREAT MODEL AND CRYPTOGRAPHIC PRIMITIVES

A.1 CROSS-SILO FEDERATED LEARNING SETTING

In standard cross-silo FL, a set of U clients (e.g., hospitals, financial institutions), denoted as $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_U\}$, collaboratively train a global model w without exchanging their private datasets D_u. In each communication round t, the clients download the global model $w^{(t)}$, perform local training steps to compute an updated local model $w_u^{(t)}$ (or gradient update), and push it back to the server. The server then aggregates these local models using a specific Aggregation Rule (AGR) to produce $w^{(t+1)}$.

SCARFL uses GHARC as its robust aggregation rule while preserving the AegisFL encrypted scalar and masked key-conversion interfaces. The system roles are the clients $\mathcal{C}$, aggregation server $\mathcal{S}_1$, and non-colluding key-holder $\mathcal{S}_2$. Clients upload one encrypted update view directly to $\mathcal{S}_1$; $\mathcal{S}_2$ assists with prescribed scalar recovery and masked key conversion without receiving the original client-uploaded ciphertexts.

A.2 ADVERSARY MODEL AND TRUST ASSUMPTIONS

We consider up to f malicious clients, with the admissible range of f determined by the selected AGR. Byzantine clients may coordinate and submit arbitrary updates. Following the three attack categories evaluated by AegisFL (Chen et al., 2024), our empirical evaluation covers an untargeted attack with Gaussian random updates, a label-flipping attack, and a scaling attack that combines triggered training samples with amplified malicious updates. Adaptive and knowledge-aware attacks remain untested. The broader threat class includes untargeted degradation attacks (Fang et al., 2020; Shejwalkar & Houmansadr, 2021; Baruch et al., 2019) and targeted model-poisoning attacks (Bagdasaryan et al., 2020; Xie et al., 2020; Wang et al., 2020).

Privacy attacks arise when a server can access plaintext intermediate results, from which sensitive information about local training data may be inferred through membership inference (Nasr et al., 2019), dataset-property inference (Zhang et al., 2021), or gradient reconstruction (Geiping et al., 2020; Zhu et al., 2019). The aggregation server $\mathcal{S}_1$ and key-holder $\mathcal{S}_2$ are *honest-but-curious* and non-colluding, as formalized in Theorem D.1: each server follows the protocol but may attempt to infer client data from its own view, while the adversary may corrupt $\mathcal{S}_1$ or $\mathcal{S}_2$ in isolation, but never both together. SCARFL assumes no trusted execution environment and no plaintext verification committee.

A.3 Cryptographic Primitives

RLWE-based homomorphic encryption. SCARFL uses the CKKS approximate homomorphic encryption scheme (Cheon et al., 2017) over the cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ for power-of-two N, with residue rings $\mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$ and level-l ciphertexts in $\mathcal{R}_{Q_l}^2$. We write $Q_l = q_0 \cdot \Delta^l$, where Δ is the CKKS scale and $0 < l \leq L$, and use the standard CKKS interfaces KeyGen, Enc, Dec, Add, cAdd, Mul, cMul, and public Galois keys GK_k for the automorphism indices used below. The equality $Q_l = q_0 \cdot \Delta^l$ is an idealized CKKS level abstraction for exposition rather than a literal modulus-chain decomposition.

Unlike standard CKKS, SCARFL uses coefficient packing rather than slot packing. Model-update vectors $m = [m_0, \ldots, m_{N-1}] \in \mathbb{R}^N$ are encoded under one of two coefficient-packing maps (Yasuda et al., 2014):

$$\mathrm{pm}_1(m, \Delta) = \Big\lfloor \Delta \sum_{i=0}^{N-1} m_i X^i \Big\rceil \in \mathcal{R}, \tag{7}$$

$$\mathrm{pm}_2(m, \Delta) = \Big\lfloor -\Delta \sum_{i=0}^{N-1} m_i X^{N-i} \Big\rceil \in \mathcal{R}. \tag{8}$$

Decoding inverts the Δ scaling and reads off the coefficient vector. The algebraic relation between pm_1 and pm_2 underlies SCARFL's single-ciphertext derivation (Theorem B.1); the arithmetic exploited by this packing is described in the practical packing paragraph below.

Galois automorphisms in cyclotomic rings. For any odd integer k coprime to $2N$, the mapping

$$\phi_k : X \longmapsto X^k \pmod{X^N + 1} \tag{9}$$

is a ring automorphism. Applying ϕ_k to a ciphertext $(c_0(X), c_1(X))$ transforms an encryption of $m(X)$ into one of $m(X^k)$ under a transformed secret key; key switching with GK_k returns it to the original key domain.

SCARFL uses the inverse-index automorphism $X \mapsto X^{-1}$. Implementations represent Galois elements by nonnegative odd integers modulo $2N$, so this automorphism is instantiated by $2N - 1$ because

$$-1 \equiv 2N - 1 \pmod{2N}. \tag{10}$$

Thus, -1 and $2N - 1$ are two representatives of the same Galois-group element.

Practical packing scheme. The two coefficient packings make encrypted scalar statistics available from the constant coefficient after one polynomial multiplication. For $a, b \in \mathbb{R}^N$, let $p_a^{(1)} \leftarrow \mathrm{pm}_1(a, \Delta)$, $p_a^{(2)} \leftarrow \mathrm{pm}_2(a, \Delta)$, and $p_b^{(2)} \leftarrow \mathrm{pm}_2(b, \Delta)$. Then

$$\begin{aligned} (p_a^{(1)} p_a^{(2)})_0 &\approx \Delta^2 \|a\|_2^2, \\ (p_a^{(1)} p_b^{(2)})_0 &\approx \Delta^2 \langle a, b \rangle. \end{aligned} \tag{11}$$

For the mean, multiplying $p_a^{(1)}$ by the fixed extractor $p = -\sum_{i=0}^{N-1} X^{N-i}$ yields $(p_a^{(1)} p)_0 = \Delta \sum_i a_i$, so $\mathrm{mean}(a) \approx (p_a^{(1)} p)_0 / (\Delta N)$. These constant-coefficient identities are the encrypted scalar primitives inherited from AegisFL and retained by SCARFL.

B Proposed Methodology

SCARFL first derives the second coefficient view from one uploaded ciphertext while retaining the encrypted norm and inner-product interface. GHARC then uses those statistics to select and weight client updates under poisoning attacks.

B.1 Single-Ciphertext Derivation via Galois Automorphism

The key redundancy in AegisFL is that each client uploads two ciphertexts, $ct^{(1)}$ and $ct^{(2)}$, encrypting two packings of the same vector. SCARFL replaces this client-side redundancy with server-side derivation. The identity applies under the same *pm coefficient-packing semantics* used by the AegisFL-style practical packing scheme, rather than under standard CKKS slot encoding.

Symbolic packing vs. scaled CKKS encoding. Throughout this section, we use pm_1 and pm_2 to denote the *symbolic* packing maps over a coefficient domain A, instantiated here as $A = \mathbb{R}$ for real-valued model updates. Thus the symbolic objects live in $A[X]/(X^N + 1)$: $\mathrm{pm}_1(\mathbf{v}) = \sum_{i=0}^{N-1} v_i X^i$ and $\mathrm{pm}_2(\mathbf{v}) = -\sum_{i=0}^{N-1} v_i X^{N-i}$. These symbolic maps are distinct from the scaled CKKS plaintext encoding $\mathrm{pm}_j(m, \Delta) = \lfloor \Delta \cdot \mathrm{pm}_j(m) \rceil$ of Section A.3. The automorphism identity holds exactly at the symbolic layer and, under sign-symmetric coefficientwise rounding, at the encoded plaintext layer. Ciphertext operations, including Galois key switching and downstream rescaling, introduce the measured CKKS approximation error.

Theorem B.1 (Exact relation between the two packings)**.** *Let A be a commutative coefficient ring, let $\mathcal{R}_A = A[X]/(X^N + 1)$ with N a power of two, and let*

$$\mathrm{pm}_1(\mathbf{v}) = \sum_{i=0}^{N-1} v_i X^i, \qquad \mathrm{pm}_2(\mathbf{v}) = -\sum_{i=0}^{N-1} v_i X^{N-i} \tag{12}$$

for any vector $\mathbf{v} = (v_0, \ldots, v_{N-1}) \in A^N$, interpreted as ring elements modulo $X^N + 1$. Then

$$\mathrm{pm}_2(\mathbf{v}) = \mathrm{pm}_1(\mathbf{v})(X^{-1}) \pmod{X^N + 1}. \tag{13}$$

The proof is the quotient-ring identity $X^{-i} = -X^{N-i}$ for $1 \le i < N$ together with $X^N \equiv -1$, so evaluating pm_1 at X^{-1} exactly reproduces the signed reverse-index polynomial pm_2. Appendix G gives the line-by-line derivation.

For coefficientwise encoding with sign-symmetric integer rounding, ϕ_{2N-1} is a signed permutation of coefficients and therefore commutes with scaling and rounding:

$$\phi_{2N-1}(\lfloor \Delta\, \mathrm{pm}_1(\mathbf{v}) \rceil) = \lfloor \Delta\, \mathrm{pm}_2(\mathbf{v}) \rceil\,. \tag{14}$$

This plaintext-encoding identity is distinct from ciphertext correctness: applying a Galois key adds key-switching noise, which is measured in Section E.3. The identity assumes the same coefficientwise rounding rule for both packings.

Proposition B.2 (Server-side derivation)**.** *Let $ct^{(1)}$ be a CKKS ciphertext under secret key $s(X)$ encrypting $\mathrm{pm}_1(\mathbf{v})$. Let ϕ_{2N-1} denote the automorphism $X \mapsto X^{2N-1} \equiv X^{-1} \pmod{X^N + 1}$, and let GK_{2N-1} be the corresponding public Galois key. Then*

$$ct^{(2)} := \mathrm{KeySwitch}(\phi_{2N-1}(ct^{(1)}), GK_{2N-1}) \tag{15}$$

is a valid ciphertext under the original secret key $s(X)$ encrypting $\mathrm{pm}_2(\mathbf{v})$ up to the standard CKKS approximation error.

Thus SCARFL reconstructs the second ciphertext required by the AegisFL-style encrypted geometric computations without asking the client to upload it; Appendix G records how this automorphism is interpreted in a concrete Galois-key interface.

B.2 Implicit Dual-View Consistency

AegisFL includes a consistency-check protocol because a malicious client can submit two independently generated ciphertexts that represent different updates. SCARFL eliminates this specific attack vector by design.

Proposition B.3 (Built-in consistency)**.** *Under SCARFL, each client uploads only one ciphertext $ct^{(1)}$. Any second ciphertext used by the protocol is generated as a deterministic public transformation of $ct^{(1)}$. Consequently, an adversarial client cannot independently choose contradictory plaintext contents for $ct^{(1)}$ and $ct^{(2)}$ within the protocol.*

This property provides *implicit dual-view consistency*: SCARFL replaces AegisFL's interactive dual-ciphertext consistency check with deterministic derivation from the first view. This establishes view consistency, not poisoning detection. If a client uploads a poisoned $ct^{(1)}$, the derived $ct^{(2)}$ is consistently poisoned as well, and detecting that poisoning remains the task of the robust aggregation rule.

Fig. 2 summarizes the substitution: the client uploads $ct^{(1)}$ directly to $\mathcal{S}_1$, which derives $ct^{(2)}$ before the encrypted scalar primitives and GHARC aggregation are evaluated.

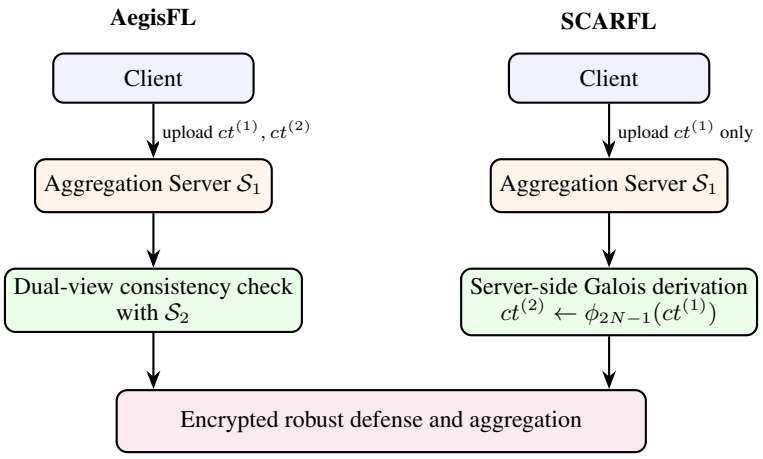

Figure 2: Processing order before robust defense. Both protocols first deliver encrypted client updates to aggregation server $\mathcal{S}_1$. AegisFL then checks the consistency of two client-supplied ciphertext views with $\mathcal{S}_2$, whereas SCARFL derives the complementary view at $\mathcal{S}_1$ through a Galois automorphism. Both paths subsequently invoke the same encrypted robust-defense interface.

B.3 Packing Semantics

The derivation above is stated under the pm coefficient-packing semantics required by the AegisFL-style practical packing scheme. The semantic interpretation of a Galois transform depends on the plaintext encoding: SCARFL applies the server-side transformation to ciphertexts encrypting $\mathrm{pm}_1(\mathbf{v})$ under coefficient packing, not to standard CKKS slot-encoded vectors. Under this packing, the inverse-index automorphism represented by $2N-1$ derives the ciphertext corresponding to $\mathrm{pm}_2(\mathbf{v})$.

B.4 GHARC Robust Aggregation

GHARC combines global-reference guidance, HDBSCAN candidate discovery, FLAME-inspired clipping evidence, cluster ranking, and adaptive reference-core expansion. Its key design choice is to separate *candidate discovery* from *candidate acceptance*: HDBSCAN identifies geometrically coherent groups, whereas a reference-guided ranking rule decides which group contributes to the global update. Algorithm 2 summarizes one aggregation round.

Historical reference and reference core. Let $\mathbf{g}_i^t$ denote the update submitted by client i at round t, and let $\bar{\mathbf{g}}^{t-1}$ be the aggregate accepted in the preceding round. An *accepted aggregate* is a nonzero aggregate whose candidate passed the clipping-mass guard; a guarded zero update is not accepted. GHARC maintains the most recent accepted aggregate as the historical reference

$$\mathbf{r}^t = \begin{cases} \bar{\mathbf{g}}^{t-1}, & \|\bar{\mathbf{g}}^{t-1}\|_2 > \epsilon, \\ \mathbf{r}^{t-1}, & \text{otherwise.} \end{cases} \tag{16}$$

At $t = 0$, before an accepted aggregate exists, the implementation uses a seed-controlled random core; the main-table runs use seed 42, and Appendix I reports seed-41 and seed-43 plaintext sweeps. Given a valid reference, GHARC computes

$$z_i^t = \cos(\mathbf{g}_i^t, \mathbf{r}^t) = \frac{\langle \mathbf{g}_i^t, \mathbf{r}^t \rangle}{\|\mathbf{g}_i^t\|_2 \|\mathbf{r}^t\|_2} \tag{17}$$

and selects the a clients with the largest z_i^t values as the reference core $\mathcal{A}^t$. A stable descending sort makes the selection deterministic when scores are tied. The evaluated configuration uses initial core size $a_0 = 7$ and recomputes the core from the current updates and historical reference.

Core-similarity representation and candidate discovery. Each client is represented by its cosine similarities to the selected core:

$$\mathbf{h}_i^t = \left[\cos(\mathbf{g}_i^t, \mathbf{g}_j^t)\right]_{j \in \mathcal{A}^t}. \tag{18}$$

The median component $s_i^t = \mathrm{median}(\mathbf{h}_i^t)$ provides a scalar core-alignment score, while the complete vector preserves the pattern of agreement with individual core members. GHARC runs HDBSCAN 0.8.44 over $\{\mathbf{h}_i^t\}_{i=1}^U$. We set `min_cluster_size`$= 2$ and use `min_samples=None`, Euclidean distance, `cluster_selection_method=eom`, and `allow_single_cluster=False`. The procedure has no randomized step. Client updates are L_2-normalized when cosine similarities are formed, but the resulting similarity feature vectors are not further standardized. Every nonoutlier group that satisfies the minimum cluster size becomes a candidate; singleton clusters are not admitted. If all labels are -1, GHARC applies a deterministic anchor-score fallback whose threshold is the maximum of the configured percentile and the median-minus-MAD cutoff; if this still selects no client, it retains the maximum-score client. A missing HDBSCAN installation or fewer than two clients invokes the deterministic adaptive-gap fallback.

FLAME-inspired clipping evidence. For each client, GHARC derives its distance from the historical reference and an adaptive clipping radius:

$$d_i^t = \|\mathbf{g}_i^t - \mathbf{r}^t\|_2, \qquad S_t = \mathrm{median}_{i\in[U]}\, d_i^t. \tag{19}$$

For a candidate cluster c, client i receives the unnormalized weight

$$\omega_{i,c}^t = \begin{cases} \dfrac{1}{|c|}\min\left(1, \dfrac{S_t}{d_i^t}\right), & i \in c, \\ 0, & i \notin c, \end{cases} \tag{20}$$

and the retained clipping mass is $W_c^t = \sum_i \omega_{i,c}^t$. Consequently, internal density alone is insufficient: a coherent but reference-distant group retains less aggregation mass.

Cluster ranking and aggregation guard. GHARC scores every eligible candidate as

$$R_c^t = \alpha W_c^t + \beta\frac{|c|}{U} + \gamma\widetilde{s}_c^t + b_c, \tag{21}$$

where $\widetilde{s}_c^t$ is the min–max normalized median core-alignment score of cluster c, and b_c is a small bonus when c is the cluster initially preferred by HDBSCAN. To define the equal-score corner case without division by zero, we use

$$\widetilde{s}_c^t = \begin{cases} 1, & s_{\max}^t - s_{\min}^t \le \epsilon, \\ \dfrac{s_c^t - s_{\min}^t}{s_{\max}^t - s_{\min}^t}, & \text{otherwise.} \end{cases} \tag{22}$$

The constant branch adds the same term to every candidate and therefore cannot alter their relative ranking. The evaluated configuration sets $\alpha = 0.75$, $\beta = 0.15$, $\gamma = 0.10$, and $b_c = 0.02$ for that initial candidate. Candidates whose clipping mass reaches the guard threshold $\tau = 0.2$ are preferred, and the highest-ranked eligible cluster c^* defines

$$\bar{\mathbf{g}}^t = \sum_{i=1}^{U} \omega_{i,c^*}^t \mathbf{g}_i^t. \tag{23}$$

The clipped weights are deliberately not renormalized after Eq. (20); their retained mass therefore controls both the direction and magnitude of the aggregate.

Adaptive reference-core expansion. A compact core reduces the number of client-to-core similarities, but a small core may not provide enough evidence in a heterogeneous round. If the guard cannot accept a candidate, GHARC increases the core size by $\Delta a = 5$, rebuilds Eq. (18), and repeats HDBSCAN, clipping, and ranking. Expansion stops when a candidate is accepted or the core reaches all U clients. If no candidate passes at $a = U$, the round returns $\bar{\mathbf{g}}^t = \mathbf{0}$, leaves the global model unchanged, and retains the previous nonzero historical reference according to Eq. (16). Section E.5 reports expansion and zero-update frequencies to test whether this guard causes stagnation.

Compatibility with the encrypted scalar interface. GHARC requires cosine similarities and Euclidean distances, both of which follow from inner products and squared L_2 norms:

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = \|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 - 2\langle \mathbf{x}, \mathbf{y}\rangle. \tag{24}$$

Algorithm 2 GHARC aggregation at round t

Input: updates $\{\mathbf{g}_i^t\}_{i=1}^U$, previous accepted reference $\mathbf{r}^{t-1}$, initial core size a_0, expansion step Δa
initialize $\mathbf{r}^t$ by Eq. (16); use a seeded random core if no valid reference exists
$a \leftarrow \min(a_0, U)$
while true **do**
  select the a clients most aligned with $\mathbf{r}^t$ as core $\mathcal{A}^t$
  construct $\mathbf{h}_i^t$ using Eq. (18) for every client
  obtain candidate clusters $\mathcal{C}$ by applying HDBSCAN to $\{\mathbf{h}_i^t\}$
  for each candidate $c \in \mathcal{C}$ **do**
    compute W_c^t, $|c|/U$, $\widetilde{s}_c^t$, and rank R_c^t
  end for
  select the highest-ranked candidate satisfying the clipping-mass guard
  if a candidate c^* is accepted **then**
    return $\bar{\mathbf{g}}^t = \sum_i \omega_{i,c^*}^t \mathbf{g}_i^t$
  end if
  if $a = U$ **then**
    return $\mathbf{0}$ and retain the previous nonzero reference
  end if
  $a \leftarrow \min(a + \Delta a, U)$
end while

The two servers evaluate these vector statistics through the encrypted primitives retained by Single-CT; HDBSCAN, cluster ranking, and guard evaluation then operate on the authorized scalar results. Server $\mathcal{S}_1$ computes each plaintext weight ω_{i,c^*}^t from those released scalars and forms $ct_{\mathrm{agg}} = \sum_i \mathrm{cMul}(ct_i, \omega_{i,c^*}^t)$ using ciphertext–plaintext multiplication and addition. This step reveals no additional update coordinate to $\mathcal{S}_1$; the weights and selected indices are functions of the already authorized scalar leakage.

C System Architecture

SCARFL uses clients, two non-colluding servers $\mathcal{S}_1$ and $\mathcal{S}_2$, and a key generation center. Clients send a single encrypted update view directly to $\mathcal{S}_1$. The server derives the complementary packing, obtains authorized norm and inner-product statistics through the dual-server interface, and executes GHARC within this direct client–server architecture.

C.1 Entities and Roles

The system contains the following entities, illustrated in Fig. 3. SCARFL adopts a standard two-server cryptographic architecture: $\mathcal{S}_1$ and $\mathcal{S}_2$ are non-colluding, and sk_s is held only by $\mathcal{S}_2$.

- **Key generation center ($\mathcal{KGC}$).** A trusted setup entity that generates and distributes public keys, secret keys, relinearization keys, and GK_{2N-1}. Server-to-client key conversion is performed online by $\mathcal{S}_1$ and $\mathcal{S}_2$ via AegisFL's masked key-conversion primitive (Chen et al., 2024), not by a preshared cross-domain key-switch key.
- **Clients ($\mathcal{C}_u$).** Data owners that perform local training, encode updates with pm_1, encrypt once, and upload only $ct_u^{(1)}$.
- **Aggregation server ($\mathcal{S}_1$).** The server that receives client ciphertexts, derives $ct_u^{(2)}$, coordinates encrypted scalar primitives with $\mathcal{S}_2$, executes GHARC on authorized scalar outputs, and holds the mask in the key-conversion primitive. $\mathcal{S}_1$ does not hold sk_s.
- **Key-holder server ($\mathcal{S}_2$).** The cryptographic peer that holds sk_s. It decrypts compressed scalar-decryption objects to release prescribed scalar statistics and performs the masked decrypt/re-encrypt steps of key conversion. $\mathcal{S}_2$ does not receive raw client ciphertexts and does not aggregate.

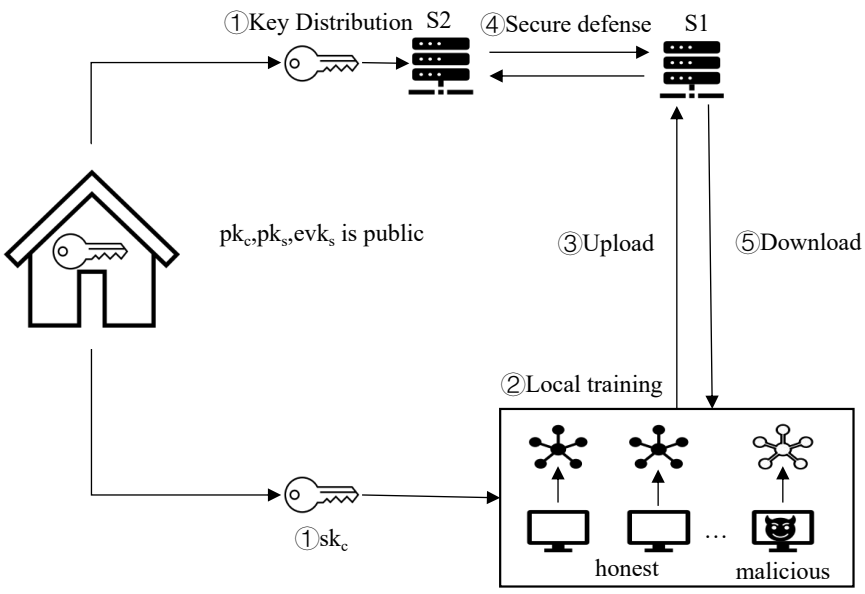

Figure 3: SCARFL system model with clients, key distribution, and two non-colluding servers. The numbered workflow comprises key distribution, local training, upload, secure defense, and model download.

C.2 WORKFLOW

SCARFL executes one training round as follows.

1. **Key Distribution:** The KGC generates $\{pk_s, sk_s, evk_s\}$ and $\{pk_c, sk_c\}$. Here evk_s consists of relinearization keys (rlk) and the required Galois keys (GK_{2N-1}). pk_s, evk_s and pk_c are broadcast to all entities in the system, sk_s is only sent to $\mathcal{S}_2$, and sk_c is only sent to clients. Server-to-client key conversion is performed online by $\mathcal{S}_1$ and $\mathcal{S}_2$ via the key-conversion primitive (detailed below), so no cross-domain key-switch key is distributed at setup time. This key partition follows the same trust boundary assumed in the baseline setting and is not introduced as an additional efficiency mechanism.

2. **Local Training:** In each communication round, clients train on local datasets to obtain gradient updates g_u, encode each update as $\mathrm{pm}_1(g_u)$, and encrypt it into $ct_u^{(1)}$ under pk_s.

3. **Upload:** Each client uploads its single ciphertext $ct_u^{(1)}$ directly to $\mathcal{S}_1$.

4. **Secure Defense:** Upon receiving $ct_u^{(1)}$, $\mathcal{S}_1$ locally derives $ct_u^{(2)}$ by applying the homomorphic automorphism $\phi : X \to X^{-1}$ and key switching. The two servers evaluate the authorized norm and inner-product statistics, $\mathcal{S}_1$ runs GHARC and aggregates the accepted updates under pk_s, and $\mathcal{S}_1$ and $\mathcal{S}_2$ finally run masked key conversion to transition the encrypted global model into the pk_c domain without exposing its plaintext.

5. **Download:** The encrypted global model under pk_c is sent directly from $\mathcal{S}_1$ to the clients, which decrypt it with sk_c for the next training round.

Key-domain transition. The step $\mathrm{KeyConvert}_{s\to c}(\cdot)$ in Algorithm 3 is the masked key-conversion primitive of §C.4. The plaintext aggregate is never exposed to either server. This cost is identical in AegisFL and SCARFL because it operates after aggregation.

Algorithm 3 SCARFL round protocol

Input: client updates $\{g_u\}_{u=1}^{U}$
Client Step:
For each client u in parallel:
compute local update g_u and encode $\mathrm{pm}_1(g_u)$ // ②
encrypt $ct_u^{(1)} \leftarrow \mathrm{Enc}(\mathrm{pm}_1(g_u), pk_s)$
upload $ct_u^{(1)}$ directly to $\mathcal{S}_1$ // ③
Server Step:
For each received $ct_u^{(1)}$, $\mathcal{S}_1$ locally derives $ct_u^{(2)} \leftarrow \mathrm{KeySwitch}(\phi_{2N-1}(ct_u^{(1)}), GK_{2N-1})$
$\mathcal{S}_1$ and $\mathcal{S}_2$ jointly evaluate the AGR-prescribed encrypted scalar statistics (L_2 norms, inner products, mean values) via the compressed-object interface (see §C.4) and obtain the scalar scores
$\mathcal{S}_1$ runs GHARC and forms encrypted global model $w_s^{(t+1)}$ under pk_s // ④
$w_c^{(t+1)} \leftarrow \mathrm{KeyConvert}_{s\to c}(w_s^{(t+1)})$ // ④
distribute encrypted $w_c^{(t+1)}$ back to clients // ⑤
each client decrypts with sk_c and starts next local round

C.3 GHARC within the Encrypted Interface

The dual-server architecture exposes encrypted norms, inner products, and means rather than update coordinates. GHARC constructs its decision from these authorized scalar outputs: similarities to core members support HDBSCAN candidate discovery and cluster ranking, while the clipping-mass guard controls whether the reference core must expand. The complementary ciphertext view is derived locally at $\mathcal{S}_1$, so SCARFL executes GHARC with only one client-uploaded ciphertext view.

C.4 Encrypted Primitive Interface

SCARFL combines the encrypted scalar-statistics and masked key-conversion interfaces inherited from AegisFL (Chen et al., 2024) with two framework components: GHARC aggregation over authorized scalar outputs and server-side derivation of the complementary ciphertext view. Describing the primitive interface here grounds the security analysis in what each server sees and what is released in plaintext.

Encrypted scalar statistics (AGR interface). For L_2-norm, inner-product, and mean-value queries, $\mathcal{S}_1$ performs a homomorphic multiplication of two complementarily packed ciphertexts $ct^{(1)}$ and $ct^{(2)}$ and extracts a *compressed scalar-decryption object* of the form $((c_0)_0,\ c_1)$, where $(c_0)_0$ is the constant coefficient of the first ciphertext component c_0 and c_1 is the full second component. This object is sent to $\mathcal{S}_2$, which decrypts the single scalar result from the constant term via $[(c_0)_0 + (sk_s \cdot c_1)_0]_{Q_l}$ and returns only that scalar. The object is roughly half the size of a full serialized ciphertext because only the constant coefficient of c_0 is transmitted (Chen et al., 2024).

Security boundary. The compressed-object format has both communication and interface-security roles: it reduces traffic by transmitting only the constant-coefficient material from c_0, and, following AegisFL's polynomial-packing design, only the decrypted constant term is treated as a semantically meaningful scalar output of the primitive (Chen et al., 2024). In our security accounting, the relevant plaintext release is therefore the returned scalar itself: a per-client norm, a pairwise inner product, or a mean value. These released scalars are explicitly included in the leakage profile $\mathcal{L}$ (see §D.2). $\mathcal{S}_1$ sees ciphertexts and the returned scalar, never plaintext updates. In SCARFL, replacing a client-uploaded $ct^{(2)}$ with an $\mathcal{S}_1$-derived one does not change this scalar-primitive interface; its privacy effect is captured by the server-side derivation hybrid below.

Server-side view derivation (SCARFL contribution). Instead of requiring clients to upload a second ciphertext $ct^{(2)}$, $\mathcal{S}_1$ computes

$$ct_u^{(2)} \;=\; \mathrm{KeySwitch}\big(\phi_{2N-1}(ct_u^{(1)}),\ GK_{2N-1}\big),$$

where $\phi_{2N-1} : X \mapsto X^{-1}$ is the ring automorphism corresponding to substitution index $2N-1$ and GK_{2N-1} is the Galois evaluation key for that automorphism. This step is purely local to $\mathcal{S}_1$, adds no

communication hop, and releases no plaintext. The result is an encryption of $\mathrm{pm}_2(g_u)$ under pk_s, which is algebraically equivalent to what a client would have uploaded in AegisFL.

Security boundary. The derivation does not require any interaction with $\mathcal{S}_2$ and introduces no new decryption event. The derived $ct_u^{(2)}$ is a public/evaluation transform of an IND-CPA ciphertext under pk_s; under IND-CPA, the joint transcript obtained by applying the public derivation to encryptions of honest updates is computationally indistinguishable from the corresponding transcript obtained by applying the same derivation to encryptions of zero. The dual-view consistency-check interaction that previously verified client-side consistency of $ct^{(1)}$ and $ct^{(2)}$ is eliminated entirely: server-side derivation enforces consistency by design.

Masked key conversion. To return the global model to clients without exposing the plaintext aggregate to any server, $\mathcal{S}_1$ and $\mathcal{S}_2$ execute the following masked exchange. $\mathcal{S}_1$ samples a fresh mask polynomial r from the plaintext mask space $\mathcal{R}_{Q_l} = \mathbb{Z}_{Q_l}[X]/(X^N + 1)$ at the current key-conversion level, masks the server-key ciphertext as $\tilde{C} = \mathrm{cAdd}(C, r)$ (adding r only to the c_0 component), and sends $\tilde{C}$ to $\mathcal{S}_2$. $\mathcal{S}_2$ decrypts $\tilde{C}$ under sk_s to obtain the ring element $r + \mathrm{Encode}_\Delta(p) + e_{\mathrm{dec}}$, where e_{dec} denotes CKKS encoding and decryption error at the current level. It re-encrypts this ring element under the client-domain key and returns the result to $\mathcal{S}_1$. $\mathcal{S}_1$ removes the mask homomorphically: $C' \leftarrow \mathrm{cAdd}(\tilde{C}', -r)$, yielding the client-decryptable ciphertext $C' = ct^{(1)}_{pk_c}(p)$.

Security boundary. The uniformity claim is made over the finite plaintext ring $\mathcal{R}_{Q_l}$, not over decoded real vectors. Conditioned on the encoded plaintext and CKKS error, adding an independently uniform $r \leftarrow_R \mathcal{R}_{Q_l}$ yields a uniform ring element. Thus $\mathcal{S}_2$ observes $r + \mathrm{Encode}_\Delta(p) + e_{\mathrm{dec}}$ but not r, while $\mathcal{S}_1$ performs unmasking homomorphically without holding sk_c. Correctness after decoding is approximate, as usual for CKKS, and its measured error is reported in Section E.3. The only backbone hops are the two full-ciphertext transfers $\mathcal{S}_1 \to \mathcal{S}_2$ (masked aggregate) and $\mathcal{S}_2 \to \mathcal{S}_1$ (re-encrypted return); all other steps are local.

D Security Analysis

D.1 Threat Model and Security Goal

We state the role-specific outputs and analyze the incremental leakage of Single-CT under static corruption. The analysis does not provide a complete real/ideal simulation for the inherited interactive primitives. Let U denote the number of participating clients in one communication round, and let $\mathcal{H}$ and $\mathcal{C}$ be the sets of honest and corrupted clients, respectively, with $|\mathcal{C}| = f$.

Adversary Model. We consider a Probabilistic Polynomial-Time (PPT) adversary $\mathcal{A}$ that can statically corrupt, in an honest-but-curious manner for server roles and with arbitrary Byzantine behavior for clients: (i) up to f clients, with $f < U/2$ in the evaluated threat model. This bound defines the tested adversarial range. Our experiments evaluate $f \in \{3, 9, 12, 14\}$ among $U = 30$ clients, reaching $f/U = 14/30 = 46.7\%$; GHARC's effective tolerance is established empirically for the attacks and configurations in Section E.4. FedAvg serves as a non-robust baseline; and (ii) either aggregation server $\mathcal{S}_1$ or key-holder $\mathcal{S}_2$ in isolation, but never both servers jointly. Only $\mathcal{S}_2$ holds the server secret key sk_s, whereas $\mathcal{S}_1$ receives client ciphertexts, derives $ct^{(2)}$, and holds the key-conversion mask. Their joint corruption is excluded because it would combine ciphertext access and masking state with decryption capability.

Non-Collusion Assumption. The privacy of SCARFL relies on the standard non-collusion assumption used in two-server cryptographic architectures: $\mathcal{S}_1$ and $\mathcal{S}_2$ do not collude. Their collusion would combine the random mask r held by $\mathcal{S}_1$ with the decrypted masked plaintext $r + p$ obtained by $\mathcal{S}_2$, revealing the plaintext aggregate p; it would also allow individual client ciphertexts visible to $\mathcal{S}_1$ to be decrypted under sk_s.

Cryptographic Assumption. The underlying CKKS homomorphic encryption scheme is IND-CPA secure under the standard Ring Learning with Errors (RLWE) hardness assumption for the chosen parameters.

Security Objective. Our protocol-level claim is that server-side ciphertext-view derivation opens no new plaintext channel beyond the inherited scalar and key-conversion interfaces. A full simulation proof for those inherited interfaces is not established here.

D.2 Party-Wise Leakage Functions and Ideal Functionality

Let $\mathbf{G} = \{\mathbf{g}_u\}_{u=1}^{U}$ denote all client updates in round t. We define leakage separately for each of the two server roles:

$$\mathcal{L}_{\mathcal{S}_1}(\mathbf{G}) = \big(t,\ U,\ f,\ \texttt{cid-set},\ \texttt{tensor-shape},\ \texttt{sim-scores},\ \texttt{final-output}\big),$$

where `final-output` is the ciphertext under pk_c. The mask r is $\mathcal{S}_1$'s own local random coin, not an input-dependent leakage item. $\mathcal{S}_1$ does not receive plaintext client updates or the plaintext global aggregate in its server role.

$$\mathcal{L}_{\mathcal{S}_2}(\mathbf{G}) = \big(t,\ U,\ f,\ \texttt{cid-set},\ \texttt{tensor-shape},\ \texttt{sim-scores},\ \texttt{masked-plaintexts}\big),$$

where `sim-scores` are the AGR-prescribed plaintext scalars obtained by decrypting the encrypted scalar-primitive compressed decryption objects under sk_s, and `masked-plaintexts` are the values $r + p$ obtained in step (iv) of the key-conversion primitive. In the full-domain mask model above, r is sampled uniformly from $\mathcal{R}_{Q_l}$ independently by $\mathcal{S}_1$ and is not revealed to $\mathcal{S}_2$ under the non-collusion assumption; hence $r + p$ is statistically uniform over $\mathcal{R}_{Q_l}$. The server-only leakage of $\mathcal{S}_2$ excludes the unmasked return. A corrupted client can decrypt the final model with sk_c; confidentiality against a server–client coalition is outside the theorem's scope.

Let $\mathcal{L} = (\mathcal{L}_{\mathcal{S}_1}, \mathcal{L}_{\mathcal{S}_2})$. Define $\mathcal{F}^{\mathcal{L}}_{\mathrm{PBFL}}$ as the ideal functionality that: (1) receives all client inputs (honest inputs from $\mathcal{H}$ and adversarial inputs from $\mathcal{C}$), (2) computes the same Byzantine-robust AGR as the real protocol, (3) returns the final model to clients and exposes the stated role-specific outputs when a server is corrupted in isolation. A corrupted client's view also includes its own input and the client-decryptable final model.

D.3 Protocol-Level Leakage Boundary

Scope of Privacy Claims. The analysis covers the incremental effect of Single-CT under static, honest-but-curious, non-colluding server roles and the declared leakage profile. The scalar-decryption and masked key-conversion primitives remain inherited assumptions; the following statement does not prove their concrete composition.

Theorem D.1 (No New Plaintext Channel from View Derivation)**.** *Let $ct^{(1)}$ be the uploaded ciphertext and $D(ct^{(1)}, GK_{2N-1})$ the public Galois derivation. Conditioned on $ct^{(1)}$ and the public evaluation key, the locally derived second view contains no additional input-dependent information for $\mathcal{S}_1$. The derivation sends no message to $\mathcal{S}_2$ and introduces no additional plaintext decryption result. Thus, when the inherited scalar and key-conversion interfaces are securely implemented with the stated outputs, Single-CT introduces no new plaintext-release channel beyond those interfaces.*

D.4 Argument and Limitation

The derived view is a deterministic polynomial-time function of material already held by $\mathcal{S}_1$, so its conditional information about an honest update is zero once the uploaded ciphertext and evaluation key are fixed. $\mathcal{S}_2$ receives neither that view nor an extra decryption request solely to derive it. The prescribed scalar outputs remain explicit leakage, and the masked conversion retains its original non-collusion requirement. A complete real/ideal simulation would need to replace honest ciphertexts with encryptions of zero while preserving the joint distribution of true scalar scores, compressed ciphertext objects, and all permitted queries; IND-CPA alone does not provide that transcript-consistency argument. Appendix H details the scope of the relative argument.

D.5 Interpretation and Boundary of Guarantees

Theorem D.1 establishes that local Galois derivation creates no additional plaintext channel. Confidentiality of the entire interactive protocol and the information-theoretic harmlessness of released similarity scores remain outside its scope. From a robustness perspective, SCARFL combines GHARC with a single-view submission interface. GHARC selects and weights updates using the current round's reference geometry, while single-view submission removes the independent second ciphertext that motivated AegisFL's consistency exchange. If the non-collusion assumption is violated, the privacy guarantee no longer holds: $\mathcal{S}_1 \leftrightarrow \mathcal{S}_2$ collusion combines ciphertext access and the key-conversion mask with decryption capability, exposing client updates or the plaintext aggregate. This is the standard limitation of the inherited dual-server PBFL model.

Repeated-round leakage boundary. The released scalar statistics (`sim-scores`) are part of the AegisFL-style robust-aggregation interface and are therefore included in $\mathcal{L}$. SCARFL does not introduce an additional plaintext score channel beyond this inherited interface. The theorem is a per-round, relative interface statement; cumulative leakage from the released scalar sequence under adaptive analysis and a complete simulation proof for the inherited interactive primitives remain open questions.

E Experiments

E.1 Experimental setup

Our prototype runs on a commodity desktop with a 64-bit Windows 10 operating system, an Intel Core i5-11400 processor (6 cores and 12 logical threads at 2.60 GHz), and 32 GB RAM. All reported experiments are CPU-only; the software environment uses Python 3.12.13 and PyTorch 2.12.1+cpu, with no CUDA acceleration. We use a Python interface to a modified Microsoft SEAL backend with the `pm` coefficient-packing mode, so the measured transformation is applied to pm-encoded rather than standard slot-encoded CKKS ciphertexts. Unless otherwise stated, we match the AegisFL-style CKKS scale: polynomial modulus degree $N = 8192$, scaling factor $\Delta = 2^{40}$, and a 200-bit modulus chain, within the standard SEAL 128-bit parameter envelope for this degree (Albrecht et al., 2022). This setting supports the per-round ciphertext depth for inner-product, L_2-norm, mean, and key-conversion primitives without refresh; each training round uses fresh client encryption, so bootstrapping is not invoked.

We evaluate 30 clients on HAR and MNIST using the lightweight neural models described in the experimental setup, and report communication accounting, encrypted-primitive noise, classification accuracy, and attack success rate under the evaluated poisoning attacks.

The evaluation addresses five questions: whether Single-CT preserves the encrypted scalar interface; how much recurrent protocol traffic it removes; what numerical and runtime cost the Galois derivation adds; whether GHARC is empirically effective under the evaluated attacks; and which GHARC components prevent either attack acceptance or training stagnation. MNIST uses an IID partition and HAR uses a subject-level Non-IID partition. Primary comparison cells use seed 42; the seed-43 plaintext GHARC sweep also appears in the main text, and Appendix I reports the seed-41 and seed-43 sweeps. The two datasets and limited seeds define the evaluated scope; broader image benchmarks, multiple seeds for all methods, stronger synthetic heterogeneity, and adaptive attacks remain untested.

SCARFL preserves the AegisFL-style encrypted scalar and masked key-conversion interfaces while using GHARC for robust aggregation and changing ciphertext-view construction. We retain AegisFL as the controlled cryptographic baseline under matched CKKS parameters, threat model, datasets, client counts, and attack settings. A non-encrypted plaintext reference in Section E.4 isolates aggregation behavior without encryption-induced perturbation.

Shared HE operators are identical after ciphertext-view construction; the operator-level error gap of SCARFL's single ϕ_{2N-1} Galois automorphism step is quantified in Section E.3. Its runtime is included in $\mathcal{S}_1$'s local HE computation and reported separately in the runtime breakdown.

Measurement methodology. Communication results are protocol communication volume obtained from the actual serialized ciphertext-object sizes; they do not emulate network latency or bandwidth.

One-time key generation and Galois-key distribution are excluded from per-round totals. Runtime uses wall-clock instrumentation around the Python orchestration and CPU cryptographic calls, includes encryption and serialization work assigned to each stage, and excludes the separately logged initialization/warm-up phase. No GPU is used and no thread affinity is pinned; HDBSCAN uses its installed default of four core-distance jobs. Both Dual-CT and Single-CT use the same process-level measurement path and serialization configuration, so paired differences isolate ciphertext-view construction. Reported means aggregate completed training rounds and the settings stated with each table; they are CPU measurements under the stated process configuration.

E.2 Communication efficiency

The primary measured benefit of SCARFL is communication reduction. A client uploads one ciphertext instead of two, so the per-round client uplink is reduced by 50% relative to AegisFL under matched CKKS parameters. The extra Galois key required for server-side derivation is a one-time setup artifact distributed during key setup rather than a per-round transmission. We instrument both pipelines end-to-end and report the per-round byte-level decomposition of the recurring saving and its protocol channels.

Protocol-level saving decomposition. To separate the cryptographic protocol from the aggregation rule, we denote the client-supplied two-view construction as **Dual-CT** and the server-derived one-view construction as **Single-CT**. These labels describe only ciphertext-view transport: Dual-CT corresponds to uploading independent pm_1 and pm_2 ciphertexts and checking their consistency, whereas Single-CT uploads only pm_1 and derives pm_2 at $\mathcal{S}_1$. We instrument both protocols and record serialized full ciphertexts and compressed scalar-decryption objects crossing each logical channel during one federated round.

The measurement covers four aggregation rules spanning the homomorphic workload. FedAvg carries no robust-statistics traffic. ShieldFL issues $3U$ scalar-result ciphertexts per non-bootstrap round for self norms, previous-global inner products, and reference-client inner products. M-FLAME materializes the upper-triangular pairwise Gram matrix together with previous-global inner products. GHARC with $U = 30$ and initial core size $a = 7$ releases 243 encrypted scalar results in a non-expansion round: one reference norm, 30 client norms, 30 client-to-reference inner products, and $30 \cdot 7 - 7 - \binom{7}{2} = 182$ distinct client-to-core inner products after self-pairs and symmetric core–core duplicates are removed. The two model sizes are $d = 72{,}710$ for HAR and $d = 101{,}770$ for MNIST, giving $n_{\mathrm{c}} = 9$ and 13 ciphertext chunks per encrypted update respectively. The Dual-CT consistency channel samples a random mask r, forms the scalar projections $\mathsf{ct}_1 \cdot \mathrm{pm}_2(r)$ and $\mathsf{ct}_2 \cdot \mathrm{pm}_1(r)$, and forwards two compressed objects to the secret-key holder. GHARC accounting uses the evaluated initial-core path; a round that triggers adaptive core expansion incurs additional scalar-statistics traffic.

Amortization scope. We distinguish per-client traffic from centrally executed per-round exchanges. Upload, Consistency, and Down. are incurred for each client; the table reports them in MB per client. Statistics and inter-server Key-conv are reported as system-wide MB per round. The key-conversion pair transfers one masked aggregate and its re-encryption per round, while the *number* of scalar-statistics requests depends on U and the AGR (quadratically in U for full pairwise geometry). Thus "system-wide" does not mean independent of U. Table 12 groups channels by accounting scope; the client-facing broadcast of the converted model is included under per-client Down..

Table 12 decomposes per-round traffic into five protocol-level channels. In this serialization format, the removed Upload and Consistency channels each contribute approximately $n_{\mathrm{c}} B_{\mathrm{ct}}$: 3.01 MB per client on HAR ($n_{\mathrm{c}} = 9$) and 4.35 MB on MNIST ($n_{\mathrm{c}} = 13$). The consistency channel actually comprises two compressed objects per chunk, not one full ciphertext. Single-CT removes the client-supplied ct_2 upload and the consistency relay. With B_{ct} the serialized full-ciphertext size and B_{comp} the serialized size of one compressed consistency object, the expected accounting is

$$\mathrm{Savings}(\textsf{Dual-CT} \to \textsf{Single-CT}) \approx U\, n_{\mathrm{c}} (B_{\mathrm{ct}} + 2B_{\mathrm{comp}}), \tag{25}$$

Here $B_{\mathrm{ct}} \approx 0.334$ MB and $2B_{\mathrm{comp}} \approx B_{\mathrm{ct}}$ only for the measured compressed-object serialization. The exact byte saving is the difference between the serialized channel totals, 180.52 MB on HAR and 260.75 MB on MNIST for GHARC. At fixed chunk count and object sizes, the two eliminated

Table 12: Per-round communication by accounting scope. Dual-CT uploads two client-supplied views; Single-CT uploads one and derives the complementary view at $\mathcal{S}_1$. Per-client channels are multiplied by $U = 30$; system-wide statistics volume can also depend on U. $\mathrm{Total} = U(\mathsf{Upload} + \mathsf{Cons.} + \mathsf{Down.}) + \mathsf{Stat.} + \mathsf{Key\text{-}conv}$. Totals use unrounded channel measurements; displayed components are rounded.

			Per-client recurring (MB/client, $\times U$=30)			System-wide per-round (MB)		Total (MB)
Dataset	AGR	Protocol	Upload	Cons.	Down.	Stat.	Key-conv	
HAR	FedAvg	Dual-CT	6.02	3.01	2.12	0.00	4.24	338.55
		Single-CT	3.01	0.00	2.12	0.00	4.24	158.03
	ShieldFL	Dual-CT	6.02	3.01	2.12	30.09	4.24	368.64
		Single-CT	3.01	0.00	2.12	30.09	4.24	188.12
	M-FLAME	Dual-CT	6.02	3.01	2.12	160.45	4.24	498.99
		Single-CT	3.01	0.00	2.12	160.45	4.24	318.49
	GHARC	Dual-CT	6.02	3.01	2.12	81.24	4.24	419.79
		Single-CT	3.01	0.00	2.12	81.24	4.24	239.27
MNIST	FedAvg	Dual-CT	8.69	4.35	3.06	0.00	6.12	489.02
		Single-CT	4.35	0.00	3.06	0.00	6.12	228.27
	ShieldFL	Dual-CT	8.69	4.35	3.06	30.09	6.12	519.11
		Single-CT	4.35	0.00	3.06	30.09	6.12	258.35
	M-FLAME	Dual-CT	8.69	4.35	3.06	160.45	6.12	649.47
		Single-CT	4.35	0.00	3.06	160.45	6.12	388.72
	GHARC	Dual-CT	8.69	4.35	3.06	81.24	6.12	570.26
		Single-CT	4.35	0.00	3.06	81.24	6.12	309.51

channels scale linearly with U; the absolute saving at $U = 100$ would be approximately $100/30$ times that at $U = 30$, but statistics traffic would also change with U. For GHARC the measured total per-round reductions are 43.0% on HAR and 45.7% on MNIST, distinct from the 50% reduction in client ciphertext upload.

Figure 4 renders the eight dataset/AGR cells as stacked bars: Statistics, Key-conv, and Down. are identical between protocols, while the saving is carried by the halved Upload segment and the removed Consistency segment. The client-decrypted key-conversion return ciphertext matches the plaintext expected aggregate $\sum_{u=1}^{U} \alpha_u g_u$, where α_u is the AGR weight actually applied in that round (not renormalized for full GHARC), to within the end-to-end pipeline noise envelope reported in Section E.3.

Beyond the empirically measured client set ($U = 30$), the same closed form extrapolates structurally along the model-dimension axis. At fixed CKKS slot capacity, the per-client chunk count n_c grows linearly with model dimension d, so scaling d by $10\times$ propagates the same proportional effect to the absolute saving as the U-linearity above. Other protocol components reshape the fractional saving rather than the absolute one: pairwise-similarity statistics used by M-FLAME carry U^2-order encrypted-statistics traffic in the system-wide half, diluting the per-client recurring saving relative to total round traffic as U grows.

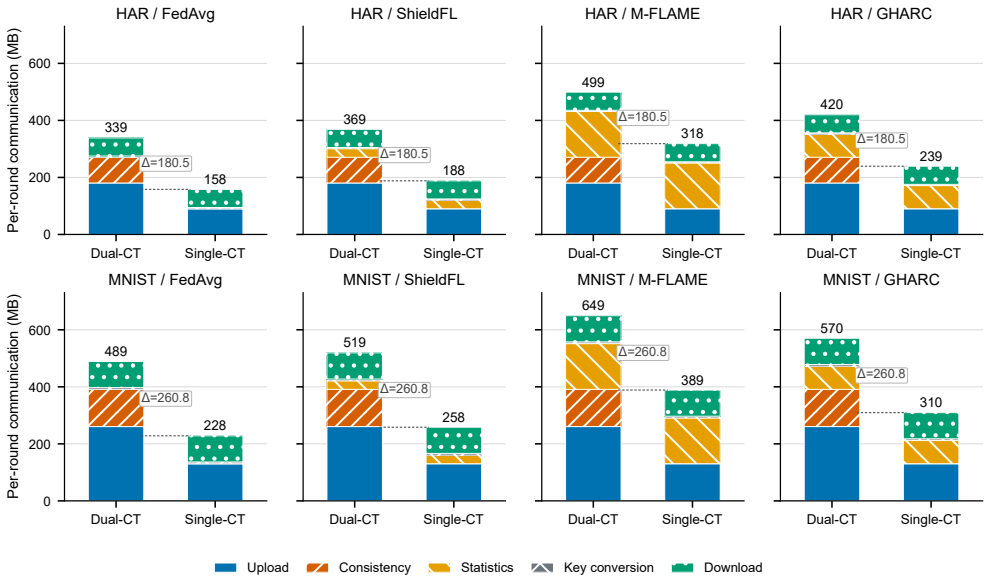

Figure 4: Per-round communication (MB) by protocol channel. Each subplot is a dataset/AGR cell; Δ marks the Single-CT saving over Dual-CT. Statistics, Key-conv, and Down. match across protocols, while Upload halves and Consistency disappears.

E.3 Numerical Accuracy and Noise Impact

Server-side derivation introduces an additional Galois automorphism and key-switching step compared with directly receiving the second ciphertext from the client. Under CKKS, this extra transformation introduces additional approximation noise. To quantify its practical impact we compare the numerical error of the encrypted inner-product primitive under Dual-CT (client-encrypted pm_2 view) and Single-CT (server-derived pm_2 view) on both the MNIST and HAR tasks. The architectural difference between the two protocols is exactly one ϕ_{2N-1} Galois automorphism; all downstream ciphertext operations are shared and therefore do not contribute to the protocol-level error gap measured here. For each dataset we sample 30 model-gradient vectors at the corresponding model dimension, normalize them to $\|v\|_2 = 5$, encrypt under the CKKS parameters of Section E.1, and compare the decrypted homomorphic inner product against the plaintext reference. The measurement is repeated over three random seeds, and we separately evaluate diagonal inner products $\langle v_i, v_i \rangle$ and off-diagonal pairs $\langle u_i, u_j \rangle$ ($i \neq j$, 50 random pairs per cell).

Table 13 reports the mean absolute error (MAE) and maximum error under both protocols and pair types. Single-CT exhibits slightly larger numerical error because the server-side automorphism adds one noise-accumulating step. Across the displayed measurement (960 inner-product samples), its worst-case maximum error is 2.17×10^{-9}, attained on HAR in the diagonal case, while the Dual-CT maximum is 5.03×10^{-10}, attained on MNIST in the off-diagonal case. From the displayed aggregates, the Single-CT/Dual-CT ratio is 4.1–$6.4\times$ by maximum error and 5.1–$6.3\times$ by MAE. The absolute extra error never exceeds 2.1×10^{-9} and remains well below the correctness threshold of 10^{-3}, so server-side derivation does not perturb the encrypted geometric primitive at decision-relevant precision.

This primitive-level measurement isolates the noise introduced by Single-CT's server-side Galois derivation. Across all twenty-four measured end-to-end rounds, the maximum drift between the client-decrypted aggregate and the plaintext reference $\sum_{u=1}^{U} \alpha_u g_u$ is 2.95×10^{-9}, comparable to the primitive-level 2.17×10^{-9} above and well below the 10^{-3} correctness threshold. The Dual-CT-only consistency interaction is absent from the Single-CT error path; the encrypted scalar and masked key-conversion primitives are otherwise shared.

Table 13: Numerical error of the encrypted inner-product primitive. "Self" denotes diagonal pairs (90 samples/cell); "Cross" denotes off-diagonal pairs (150 samples/cell). MAE and Max. are aggregated across three seeds.

Dataset	Protocol	Pair	MAE	Max. error
MNIST	Dual-CT	Self	1.12×10^{-10}	3.28×10^{-10}
		Cross	9.74×10^{-11}	5.03×10^{-10}
	Single-CT	Self	6.29×10^{-10}	2.10×10^{-9}
		Cross	6.15×10^{-10}	2.08×10^{-9}
HAR	Dual-CT	Self	1.13×10^{-10}	3.98×10^{-10}
		Cross	1.04×10^{-10}	3.05×10^{-10}
	Single-CT	Self	5.79×10^{-10}	2.17×10^{-9}
		Cross	5.36×10^{-10}	1.74×10^{-9}

E.4 Robustness Evaluation

We evaluate GHARC against FedAvg, M-FLAME, and ShieldFL on MNIST and HAR. The GHARC Plain rows use the completed seed-42 sweep. The GHARC runs have no separate clean FedAvg initialization round before the measured, attack-aware training rounds. Every reported run uses 30 clients, 2,000 measured global rounds, 10 local iterations, local learning rate 0.03, and server learning rate 0.5. We vary the malicious-client count over $\{3, 9, 12, 14\}$ and evaluate the AegisFL-style scaling, label-flipping, and untargeted (UT) attacks (Chen et al., 2024) in plaintext and Single-CT homomorphic-encryption modes. In the scaling attack, malicious clients add triggered copies of 20% of their local samples, zero every 20th feature, assign target label 2, and multiply their submitted updates by 5. Label flipping maps label ℓ to $M - 1 - \ell$ for M classes; UT replaces malicious updates with Gaussian noise. The initial reference-core size is seven. For each run, we average accuracy and, where applicable, ASR over rounds 1990–1999. Tables 14 and 15 report every malicious-client setting separately. Scaling entries are clean accuracy (Acc) / triggered-sample attack success rate (ASR); label-flipping and UT entries are Acc.

Table 14: Robustness comparison on HAR. The GHARC row uses the initialization stated in the text. Each cell is the last-10-round mean in percent. Scaling reports clean Acc/triggered-sample ASR; Flipping and UT report Acc. Column labels denote malicious clients out of 30.

AGR	Mode	Scaling (Acc/ASR)				Flipping (Acc)				UT (Acc)			
		3	9	12	14	3	9	12	14	3	9	12	14
FedAvg	Plain	97.10/97.81	97.08/99.62	97.27/99.43	97.74/99.67	94.39	80.61	69.44	60.56	91.20	88.93	86.88	86.65
	Single-CT	97.11/97.81	97.08/99.62	97.23/99.43	97.78/99.67	94.40	80.61	69.36	57.97	91.21	88.93	86.91	86.64
M-FLAME	Plain	96.72/0.00	96.04/99.29	96.17/99.28	96.37/99.62	96.69	96.46	96.52	95.65	96.64	87.30	87.91	87.05
	Single-CT	96.77/0.00	95.96/99.18	96.17/99.33	96.27/99.62	96.79	96.46	96.53	95.63	96.74	87.31	87.91	87.05
ShieldFL	Plain	95.63/97.92	96.49/99.48	96.42/99.44	95.25/99.55	89.22	55.60	40.35	58.78	91.47	92.84	92.12	92.19
	Single-CT	95.20/97.67	95.69/99.54	95.27/99.47	95.88/99.59	90.13	68.80	45.47	36.38	92.03	92.55	92.73	92.17
GHARC	Plain	96.25/0.14	95.77/0.09	95.78/0.05	95.55/0.05	96.33	95.80	95.78	95.52	95.98	95.88	95.83	95.33
	Single-CT	96.08/0.14	95.91/0.04	95.57/0.05	95.70/0.05	96.42	95.89	95.81	95.56	95.95	95.79	95.61	95.52

Table 15: Robustness comparison on MNIST. The GHARC row uses the initialization stated in the text. Each cell is the last-10-round mean in percent. Scaling reports clean Acc/triggered-sample ASR; Flipping and UT report Acc. Column labels denote malicious clients out of 30.

AGR	Mode	Scaling (Acc/ASR)				Flipping (Acc)				UT (Acc)			
		3	9	12	14	3	9	12	14	3	9	12	14
FedAvg	Plain	95.96/96.35	96.73/98.72	97.00/99.06	97.12/99.17	96.03	94.53	89.31	68.57	84.95	83.64	83.17	82.89
	Single-CT	95.96/96.35	96.73/98.72	97.00/99.06	97.12/99.17	96.03	94.53	89.31	68.57	84.95	83.65	83.17	82.89
M-FLAME	Plain	95.66/0.67	95.49/0.52	95.43/0.50	95.45/0.49	95.72	95.49	95.43	95.45	95.70	95.04	94.33	91.50
	Single-CT	95.66/0.65	95.49/0.52	95.43/0.50	95.45/0.49	95.72	95.49	95.43	95.45	95.70	95.04	94.33	91.50
ShieldFL	Plain	97.84/96.04	97.81/98.92	97.80/99.32	97.87/99.30	57.68	45.17	48.51	47.48	97.77	97.67	97.12	97.08
	Single-CT	97.75/96.38	97.77/98.67	97.82/98.11	97.73/99.39	55.79	48.12	48.30	47.46	97.82	97.60	97.14	97.07
GHARC	Plain	95.48/0.53	95.47/0.55	95.58/0.56	95.49/0.60	95.56	95.51	95.52	95.41	95.52	95.49	95.24	95.25
	Single-CT	95.65/0.51	95.32/0.54	95.21/0.54	95.33/0.48	95.60	95.42	95.43	95.35	95.55	95.40	95.29	95.02

Result. Across the 48 reported GHARC configurations, last-10-round mean accuracy remains between 95.02% and 96.42%. Under the scaling attack, GHARC has triggered-sample ASR at or below 0.14% on HAR and 0.60% on MNIST, including $14/30$ malicious clients. FedAvg and ShieldFL have ASR between 96.04% and 99.67% in these cells; M-FLAME suppresses the scaling attack on HAR at three malicious clients but its ASR exceeds 99% at nine, twelve, and fourteen malicious clients. Under label flipping, GHARC remains between 95.35% and 96.42% Acc; under UT, between 95.02% and 95.98%. ShieldFL obtains higher MNIST UT accuracy. These results characterize the evaluated attacks; they do not establish a Byzantine-tolerance theorem.

Appendix I reports seed-41 and seed-43 plaintext GHARC sweeps under the same nominal datasets, partition types, attacks, client counts, learning rates, and round budget as the seed-42 GHARC Plain rows. Across all three plaintext seeds, scaling-attack ASR is at most 0.40% on HAR and 0.73% on MNIST. These sweeps support consistency under the tested settings but are not statistically sufficient evidence of general robustness.

Appendix J compares Dual-CT and Single-CT GHARC robustness for all $f \in \{3, 9, 12, 14\}$ encrypted runs.

E.5 GHARC Component Ablation

We isolate the GHARC decision path under the most adversarial evaluated scaling-attack setting, $f = 14$ of $U = 30$. All variants use plaintext execution, 2,000 rounds, 10 local iterations, local learning rate 0.03, server learning rate 0.5, initial core size seven, seed 42, IID MNIST, and subject-partitioned HAR. Table 16 reports mean accuracy and ASR over rounds 1990–1999, matching the main robustness tables. The full-GHARC ablation trajectory is identical, round for round in accuracy and ASR, to the replacement seed-42 GHARC Plain scaling-attack run at $f = 14$ on each dataset; it reuses the same baseline. Each variant changes one decision-path component relative to full GHARC while retaining the historical-reference rule and numerical hyperparameters. Without HDBSCAN, all clients form a single candidate cluster; without clipping, within-cluster weights are uniform; without ranking, the initial HDBSCAN candidate is retained; without adaptive expansion, the unchanged guard is evaluated only at the initial core size. The normalized-weight variant changes only the final post-selection weight scaling. The resulting guard behavior is part of each component's effect, not an additional independently disabled component.

Table 16: One-factor-at-a-time GHARC ablation under the scaling attack with $14/30$ malicious clients. Clean Acc/triggered-sample ASR are final-10-round means in percent. Zero is the number of guarded zero-update rounds over all 2,000 rounds; Expand counts rounds that increased the initial core. A dash denotes a deliberately fixed core.

	HAR, subject Non-IID				MNIST, IID			
Variant	Acc	ASR	Zero	Expand	Acc	ASR	Zero	Expand
Full GHARC	95.55	0.05	0	0	95.49	0.60	0	32
GHARC w/o HDBSCAN	96.53	99.62	0	0	95.52	98.58	0	0
GHARC w/o clipping	95.37	99.67	0	0	96.54	99.28	0	0
GHARC w/o cluster ranking	96.64	99.62	0	0	95.47	0.65	0	23
GHARC w/o adaptive expansion	95.55	0.05	0	–	90.49	0.98	1788	–
GHARC with normalized weights	95.55	0.05	0	0	95.48	0.58	0	10

Replacing HDBSCAN by a single all-client candidate or removing clipping preserves clean accuracy but raises ASR to 98.58–99.67% on both datasets. The higher MNIST accuracy without clipping does not indicate a stronger defense because scaling-attack updates are almost always accepted. HDBSCAN supplies the candidate partition, while clipping supplies historical-reference evidence used to reject a coherent but poisoned candidate.

Cluster ranking has a dataset-dependent role. On IID MNIST, retaining the initial HDBSCAN candidate changes final-10-round ASR from 0.60% to 0.65%. On subject-partitioned HAR, it raises ASR from 0.05% to 99.62% while accuracy remains high. Ranking is therefore necessary in the

evaluated heterogeneous setting, where HDBSCAN can expose several plausible groups and the initially selected group need not be the reference-consistent one. Retained clipping mass captures proximity to the historical reference, while core-alignment score captures the pattern of similarity to selected core members; these are related but not identical signals.

Adaptive expansion is conditional rather than continuously active. HAR never invokes it, so the fixed-core and full rows coincide. MNIST invokes expansion in 32 of 2,000 rounds: 31 rounds terminate at core size 12 and one at size 17. Disabling expansion causes 1,788 guarded zero updates, including all of the final 10 rounds, and lowers final-10-round accuracy to 90.49%. Full GHARC has no zero-update round. Renormalizing accepted weights changes final-10-round accuracy by 0.01 percentage point and ASR by 0.02 percentage point, while reducing MNIST expansion from 32 to 10 rounds. The observed resistance therefore cannot be explained solely by an unnormalized reduction in update magnitude.

F Related Work

Privacy-preserving Byzantine-robust federated learning (PBFL) combines update confidentiality with poisoning resistance. Early PBFL designs pair homomorphic encryption with auxiliary robustness mechanisms, such as privacy-enhanced filtering against poisoning adversaries (Liu et al., 2021) and blockchain-assisted variants (Miao et al., 2022). More recent systems use richer encrypted-domain statistics, including cosine similarity, dynamic trust weighting, and functional-encryption primitives, to identify malicious encrypted updates under IID and non-IID settings. Pure secure-aggregation protocols in the Bonawitz line (Bonawitz et al., 2017) hide individual updates during summation, but do not expose the encrypted scalar statistics needed by similarity-based Byzantine-robust filtering.

Encrypted Byzantine-robust statistics. A first line of recent PBFL work innovates on the encrypted detection statistic itself while keeping the cryptographic interface broadly fixed. Lancelot (Jiang et al., 2025) studies compute-efficient Byzantine-robust FL with fully homomorphic encryption and encrypted sorting; DBFL (Chen et al., 2025), BPFLH (Zhu et al., 2026), and the two-dimensional HE-assisted scheme of Gu et al. (2026) focus on encrypted distance, density, or dissimilarity mechanisms under heterogeneous or non-IID data; RobustPPFL (Xue et al., 2026) uses privacy-preserving model-performance verification and secure inference; and PEAR (Sun et al., 2025) stays within a CKKS single-key dual-server interface and builds Byzantine tolerance from encrypted cosine similarity, trust-score generation, and two-step weighting. SCARFL instead uses GHARC to construct candidate-cluster decisions from norms and inner products already supported by the AegisFL-style encrypted scalar interface.

Alternative cryptographic interfaces. A second line of recent work changes the cryptographic primitive on which the encrypted aggregation interface is built. Functional-encryption-based designs such as SAMFL (Guan et al., 2024) realize Byzantine-robust secure aggregation through functional-encryption primitives rather than a CKKS/RLWE scalar-primitive interface, and BSR-FL (Zeng et al., 2024) explores a non-interactive functional-encryption route to secure cosine-similarity-based Byzantine filtering. These systems target a different interface from AegisFL-style flexible AGRs. SCARFL remains within the CKKS/RLWE interface and derives the second view by Galois automorphism, leaving the underlying cryptographic primitive unchanged.

AegisFL interface and SCARFL contribution. AegisFL (Chen et al., 2024) is the closest cryptographic baseline: it exposes encrypted primitives for inner products, L_2 norms, and means, introduces M-FLAME as an HE-compatible modification of FLAME (Nguyen et al., 2022), and uses masked key conversion to return the aggregate to the client key domain. SCARFL preserves these interfaces while removing the second client-supplied ciphertext through Galois derivation. GHARC uses the retained interface for robust aggregation. The mechanism-level comparison is in Appendix L (Table 22).

G Single-Ciphertext Derivation Details

This appendix expands the algebraic proof and implementation interpretation behind Theorem B.1 and Proposition B.2.

Proof of Theorem B.1. In $\mathcal{R}_A$ we have $X^N \equiv -1$, hence for every $i \in \{1, \dots, N-1\}$,

$$X^{-i} = X^{N-i}X^{-N} = -X^{N-i}. \tag{26}$$

Therefore

$$\mathrm{pm}_1(\mathbf{v})(X^{-1}) = \sum_{i=0}^{N-1} v_i X^{-i} \tag{27}$$

$$= v_0 + \sum_{i=1}^{N-1} v_i X^{-i} \tag{28}$$

$$= v_0 - \sum_{i=1}^{N-1} v_i X^{N-i}. \tag{29}$$

On the other hand,

$$\mathrm{pm}_2(\mathbf{v}) = -v_0 X^N - \sum_{i=1}^{N-1} v_i X^{N-i} \equiv v_0 - \sum_{i=1}^{N-1} v_i X^{N-i} \tag{30}$$

modulo $X^N + 1$. Hence (13) holds exactly as an equality in the quotient ring.

Galois-key interface for Proposition B.2. Conceptually, the derivation consists of an automorphism followed by restoration to the original secret-key space using the corresponding Galois key. Any CKKS library exposing a Galois-key automorphism interface realizes this as a single primitive: the nonnegative representative $2N-1$ realizes the inverse-index automorphism $-1 \bmod 2N$, so the primitive realizes the required mapping $X \mapsto X^{-1}$ on ciphertexts encoded under the pm semantics. Functionally, this server-derived $ct^{(2)}$ replaces a client-generated encryption of $\mathrm{pm}_2(\mathbf{v})$ in the subsequent homomorphic inner-product and norm computations.

H Scope of the Relative Leakage Argument

Theorem D.1 compares the contribution of the second-view construction. For $\mathcal{S}_1$, let V denote the received first-view ciphertexts, public keys, and local randomness. The derived views are $D(V)$, so the transcript portion $(V, D(V))$ is computable from V alone. Any distinguisher using $(V, D(V))$ can be run on V by evaluating D locally. This is a post-processing statement, independent of whether the scalar outputs themselves reveal information.

For $\mathcal{S}_2$, the Galois derivation is local to $\mathcal{S}_1$ and requires no interaction. The scalar-statistic and masked key-conversion exchanges remain the inherited interface. A corrupted client knows sk_c and can decrypt the final model; that output is available by design and must be included when analyzing a server–client coalition. In the full-domain mask model, $r + p$ is uniform to $\mathcal{S}_2$ without r. This observation covers the mask distribution, while security of the entire key-conversion implementation under all correlations or parameter choices requires a separate analysis.

In particular, a simulation replacing honest client ciphertexts with encryptions of zero would need to account for the joint distribution of the real scalar-decryption results, compressed ciphertext objects, and all permitted queries; IND-CPA alone is insufficient. The present claim concerns the absence of an *additional* plaintext channel caused by local view derivation. Security of the inherited primitives and cumulative leakage from released scores are separate questions.

I Independent Seed-41 and Seed-43 Plaintext GHARC Sweeps

Tables 17 and 18 report every completed GHARC case for seeds 41 and 43. Each sweep uses 30 clients, 2,000 rounds, 10 local iterations, local/server learning rates $0.03/0.5$, full-batch sequential local sampling, initial core size seven, malicious-client set from the front, IID MNIST with its respective partition seed, and subject-partitioned HAR. No separate clean training round is used. Both sweeps use the same launcher configuration as the seed-42 GHARC Plain rows. Each entry

averages rounds 1990–1999, using the same reporting window as the main robustness tables. No HE run or competing AGR was executed for seeds 41 or 43.

Table 17: Additional seed-41 plaintext GHARC results (last-10-round means, percent). Scaling entries are clean Acc/triggered-sample ASR; flipping and UT entries are clean Acc. All 24 cases completed 2,000 rounds.

Dataset	Attack	$f = 3$	$f = 9$	$f = 12$	$f = 14$
HAR	Scaling (Acc/ASR)	96.17/0.31	95.76/0.40	95.29/0.39	95.03/0.39
	Flipping (Acc)	96.18	95.73	95.35	95.04
	UT (Acc)	96.09	95.79	95.36	94.98
MNIST	Scaling (Acc/ASR)	95.56/0.55	95.46/0.50	95.43/0.51	95.35/0.54
	Flipping (Acc)	95.63	95.47	95.42	95.29
	UT (Acc)	95.59	95.38	95.08	95.03

Table 18: Additional seed-43 plaintext GHARC results (last-10-round means, percent). Scaling entries are clean Acc/triggered-sample ASR; flipping and UT entries are clean Acc. All 24 cases completed 2,000 rounds.

Dataset	Attack	$f = 3$	$f = 9$	$f = 12$	$f = 14$
HAR	Scaling (Acc/ASR)	96.58/0.18	95.91/0.17	95.95/0.10	95.36/0.04
	Flipping (Acc)	96.93	95.94	96.10	95.33
	UT (Acc)	96.77	96.28	95.90	95.19
MNIST	Scaling (Acc/ASR)	95.51/0.60	95.47/0.62	95.48/0.56	95.46/0.73
	Flipping (Acc)	95.54	95.49	95.45	95.25
	UT (Acc)	95.38	95.28	95.32	95.01

J DUAL-CT GHARC ROBUSTNESS CROSS-CHECK

Table 19 compares the two ciphertext-view protocols under seed 42, 30 clients, 2,000 rounds, 10 local iterations, the three evaluated attacks, and $f \in \{3, 9, 12, 14\}$. Each entry is the mean over rounds 1990–1999. All 24 Dual-CT cases completed the full round budget. Across these cells, the largest observed absolute Single-CT/Dual-CT difference is below 0.34 percentage point in clean accuracy and 0.12 percentage point in scaling-attack ASR. This comparison concerns robustness outcomes; the separately measured protocol runtime is reported in Appendix K.

Table 19: GHARC robustness under Single-CT and Dual-CT (seed 42; last-10-round means in percent). Scaling entries are clean Acc/triggered-sample ASR; flipping and UT entries are clean Acc.

Dataset	Attack	Protocol	$f = 3$	$f = 9$	$f = 12$	$f = 14$
HAR	Scaling	Single-CT	96.08/0.14	95.91/0.04	95.57/0.05	95.70/0.05
		Dual-CT	96.26/0.18	95.86/0.08	95.74/0.07	95.57/0.05
	Flipping	Single-CT	96.42	95.89	95.81	95.56
		Dual-CT	96.37	96.10	95.64	95.39
	UT	Single-CT	95.95	95.79	95.61	95.52
		Dual-CT	96.28	95.61	95.71	95.50
MNIST	Scaling	Single-CT	95.65/0.51	95.32/0.54	95.21/0.54	95.33/0.48
		Dual-CT	95.65/0.51	95.32/0.54	95.25/0.55	95.49/0.59
	Flipping	Single-CT	95.60	95.42	95.43	95.35
		Dual-CT	95.61	95.43	95.46	95.45
	UT	Single-CT	95.55	95.40	95.29	95.02
		Dual-CT	95.55	95.35	95.33	95.22

K RUNTIME COMPARISON

We evaluate runtime under the same 30-client, local-iteration, learning-rate, attack, and malicious-client settings as the main experiments. For every dataset/AGR pair, Table 20 averages the per-round time over 3 attacks and 4 malicious-client counts (3, 9, 12, and 14), i.e., 12 runs per reported value. The GHARC rows correspond to the robust aggregation rule used by SCARFL.

Table 20: Mean runtime per training round (seconds). Plain denotes unencrypted execution; Single-CT denotes PM1 upload with server-side Galois derivation; Dual-CT denotes two client-supplied ciphertext views with consistency checking. Saving is relative to Dual-CT.

Dataset	AGR	Plain	Single-CT	Dual-CT	Saving
HAR	FedAvg	0.347	1.657	3.185	48.0%
	ShieldFL	0.357	4.438	5.429	18.2%
	M-FLAME	0.360	14.487	15.605	7.2%
	GHARC	0.379	8.121	9.120	11.0%
MNIST	FedAvg	2.511	4.372	6.535	33.1%
	ShieldFL	2.512	8.337	9.744	14.4%
	M-FLAME	2.522	22.685	24.107	5.9%
	GHARC	2.585	13.872	15.271	9.2%

Single-CT is faster than Dual-CT in every measured dataset/AGR cell. For GHARC, it reduces mean per-round time from 9.120 to 8.121 seconds on HAR (11.0%) and from 15.271 to 13.872 seconds on MNIST (9.2%). GHARC's core-restricted encrypted geometry is also less expensive than M-FLAME's broader pairwise workload: under Single-CT, GHARC is 43.9% faster on HAR and 38.9% faster on MNIST.

GHARC stage breakdown. Table 21 isolates where the protocol-level saving arises. Local is client training, Upload is encryption and upload, Cons. is the Dual-CT consistency stage, Galois is Single-CT's server-side PM1-to-PM2 derivation, HE-stat is encrypted statistic evaluation, HE-agg is encrypted weighted aggregation, and Dec./Upd. is aggregate decryption and model update.

Table 21: GHARC per-round runtime breakdown (seconds), averaged over the same 12 settings per dataset.

Dataset	Protocol	Local	Upload	Cons.	Galois	HE-stat	HE-agg	Dec./Upd.	Eval.	Total
HAR	Single-CT	0.310	1.078	0.000	0.519	6.035	0.115	0.059	0.005	8.121
	Dual-CT	0.309	2.152	0.437	0.000	6.041	0.116	0.059	0.005	9.120
MNIST	Single-CT	2.462	1.576	0.000	0.763	8.770	0.187	0.085	0.028	13.872
	Dual-CT	2.468	3.138	0.618	0.000	8.747	0.186	0.085	0.028	15.271

The runtime difference comes from the removed second-view upload and consistency check, offset by Galois derivation. On MNIST, for example, Single-CT avoids approximately 1.562 seconds of extra upload and 0.618 seconds of consistency processing while adding 0.763 seconds of Galois work. The remaining stage times are closely matched, so the comparison isolates ciphertext-view construction rather than changing GHARC itself.

Consistency-check cost. The Dual-CT consistency stage averages 0.437 seconds on HAR and 0.618 seconds on MNIST for GHARC. Its two symmetric encrypted projection paths dominate the cost. Single-CT removes both paths because its complementary view is deterministically derived from the uploaded ciphertext.

L MECHANISM COMPARISON

Table 22 contrasts AegisFL and SCARFL within the shared encrypted interface. The complementary view migrates from client upload to server-side Galois derivation, the consistency exchange is removed, and SCARFL uses GHARC over the authorized scalar outputs while preserving masked key conversion.

Table 22: Mechanism-level comparison of AegisFL and SCARFL. SCARFL preserves the encrypted scalar and masked key-conversion interfaces while combining GHARC aggregation with single-ciphertext upload.

Protocol function	AegisFL (Chen et al., 2024)	SCARFL (this work)	Structural effect
Second encrypted view $\mathrm{ct}^{(2)}$	client uploads the pm_2 view	$\mathcal{S}_1$ derives it via ϕ_{2N-1} plus key switching	removes the redundant client upload
Dual-view consistency check	interactive random-mask relay between $\mathcal{S}_1$ and $\mathcal{S}_2$	unnecessary by construction	removes one protocol interaction per round
Encrypted scalar primitives	inner product, L_2 norm, mean via scalar-decryption objects	same interface, same primitives	inherits AGR flexibility unchanged
Masked key-conversion primitive	$\mathcal{S}_1 \leftrightarrow \mathcal{S}_2$ masked round trip to the client key domain	same primitive, unchanged	preserves client-side final-model ownership
Robust aggregation	external AGR over authorized scalar outputs	GHARC core construction, HDBSCAN candidates, ranked clipping evidence, and adaptive expansion	interface-compatible poisoning defense