Available Online: 16 November 2026 DOI: 10.54254/2977-3903/2026.39213

Poisoning detection in coded federated learning: impact of linear mixed aggregation on an FLPD-style detector

Siyuan Yangye

School of Control and Computer Engineering, North China Electric Power University, Beijing, China

alphonsoatkinson518@gmail.com

Abstract. Federated Learning (FL) lets clients train a shared model without sharing raw data, but malicious clients can poison it with forged updates whose validity the central server cannot directly verify. Similarity-based detection compares client updates, while coded aggregation reduces uplink cost by mixing updates into packets, whether detection remains effective after such mixing is unclear. This paper evaluates the question on Fashion-MNIST with 100 clients, 10 of them malicious, under two coded aggregation modes and three attacks, namely label flipping, scaled sign reversal and a LIE-style attack. The first two attacks cover mixing sizes m = 1, 2, 5 and 10, the LIE-style variant covers decisive configurations. The LIE-style attack is a purpose-built variant and differs from the published algorithm. In Mode A, decoding precedes detection, which remains lossless in the tested configurations. In Mode B, detection acts on mixed packets and depends on the grouping layout. Under the LIE-style attack at m = 10, individual-level detection coverage is 100%, whereas packet-level coverage is 23% under spread and 100% under consecutive, these units differ, and spread accuracy remains at 86.03%. Under scaled sign reversal at m = 10 with spread, the defense can make the model worse than no defense. Saved traffic can therefore cost detection capability, with outcomes depending on layout as well as mixing size.

Keywords: federated learning, poisoning attack detection, coded aggregation, similarity-based detection, communication–security trade-off

1. Introduction

Federated Learning (FL) keeps raw training data on clients and aggregates their model updates at a server [1]. Keeping data local does not guarantee update validity. A malicious client can exploit this gap by forging an update, as illustrated by the original LIE attack [2].

Malicious clients can poison local data or forge updates. Some similarity-based defenses exploit the resemblance among updates submitted by colluding attackers, while non-IID data cause honest updates to differ. Applying such defenses to individual clients requires access to their separate updates.

A second research line introduces linear mixing to reduce upload cost. FedNC decodes random linear combinations into individual updates [3]. In CHFL, relays sum the updates, so the server obtains the sum rather than separate client updates [4]. Coding at the data-allocation layer has instead been reported to bring honest gradients closer and support robust aggregation [5]. These coding operations affect different parts of

Copyright: © 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

the learning process. Guarantees concerning eavesdropping or packet loss do not establish resistance to forged updates. The narrower question examined here is how similarity-based detection behaves when fixed, disjoint groups replace individual updates with sums under different mixing sizes and layouts.

This paper examines whether similarity-based poisoning detection remains valid after linear mixing. The evaluation covers two coded aggregation modes, mixing sizes from 1 to 10, two grouping layouts and three synchronous white-box attacks, with detection coverage and model accuracy measured on Fashion-MNIST for 100 clients. The aim is to identify the conditions under which bandwidth saving and detection capability can coexist.

This study treats detection resolution as a design variable alongside uplink cost. Its evaluation separates the effects of coding mode, mixing size and grouping layout, so that detection capability can be assessed together with model accuracy. The resulting evidence can inform bandwidth-saving coded aggregation schemes under the tested threat model.

2. Background and related work

2.1. Similarity-based poisoning detection

The attacks considered here include targeted label flipping, scaled sign reversal and the LIE-style attack, where LIE abbreviates the phrase "A Little Is Enough". They are defined in Section 3.1.

FLPD downweights suspicious clients, while Krum, Trimmed-Mean, FLTrust and FoolsGold address poisoning through different aggregation or similarity criteria [6-10]. The detector used in this paper is a simplified variant of FLPD, with its departures disclosed in Section 3.3. UDFed considers both data and model poisoning [11]. These approaches evaluate individual client updates, so their detection inputs differ from the packet sums examined here. Wei and Liu investigate the coupling left by poisoning itself, rather than the coupling introduced by linear mixing [12].

2.2. Coded federated learning

Network coding forwards linear combinations and can reach the upper bound of the multicast cut-set. FedNC performs linear combinations and then decodes every individual update by Gaussian elimination [3]. CHFL evaluates the sum at the relay, so only the sum reaches the server and m vectors are compacted into one [4]. The security guarantees of both schemes address eavesdropping rather than poisoning, and the same limitation applies to FedSNC and to more recent designs [13-15].

CRA-DL appears contradictory to the present study, because coding at the data-allocation layer makes honest gradients closer to each other and thus makes detection easier [5]. The coding in Mode B, however, happens at the communication layer and acts directly on the detection input. The layer at which coding happens therefore determines whether it helps or harms detection.

Cryptographic coding differs from the linear mixing of network coding. Coded computing is combined with vector commitments in ByzSecAgg, whose invertible transform changes the legibility of gradients, whereas the sum in Mode B is irreversible [16]. Gradient skewness is exploited by STRIKE to circumvent existing defenses [17].

2.3. Detection without individual gradients

Aggregate-only detection is not new. SAFELearning uses one-time random, disjoint subgroups and selectively discloses subgroup parameters to detect backdoors at group level without identifying individual clients [18].

FedGT identifies malicious clients through overlapping groups and tests on securely aggregated group models, its assignment matrix also serves as a parity-check matrix [19]. Detect & Score extends this approach across rounds [20]. The overlapping-group experiment in the present study was confounded by changes in the relative detection threshold and does not support a comparison across overlap degrees. Whisper bisects disjoint sub-batches to isolate failing proofs at $1 + log(\frac{n_b}{d})$ extra rounds [21]. Its guarantee concerns proof validity, so semantically valid poisoned updates may still pass. FedGT requires some groups to test negative. The present study examines the evaluated spread configurations in which no packet is free of malicious contributions.

LSHFed hashes each gradient into a compact binary string and detects outliers using Hamming distance [22]. ByzSecAgg combines coded computing with vector commitments to detect Byzantine clients [16]. These approaches retain a way to evaluate client contributions. They therefore motivate distinguishing per-client compression from irreversible cross-client summation, which changes the objects available to a similarity detector.

3. System model and problem definition

3.1. System model and attack model

The system consists of one server and N = 100 clients. The data are non-IID and split by a Dirichlet distribution with α = 0.5. In each round, client k downloads $w(t)$, runs E = 1 local SGD epoch, and then uploads its parameter update $\Delta_k(t) = w_k(t) - w(t)$. This is not a raw gradient, because the learning rate has been folded in. The server averages the updates as

$$w(t+1) = w(t) + (\frac{1}{N}) \times \sum_k \Delta_k(t) \tag{1}$$

Uniform weights $1/N$ are used. Textbook FedAvg weights by local data size, which differs from the formulation adopted here [1]. The choice is deliberate and is applied to every configuration.

Table 1 summarizes the three attack settings and their study-specific parameters. In targeted label flipping, all local labels become class 5, and the poisoned update is scaled by 8.85. In scaled sign reversal, the forged update points opposite to the honest mean and is amplified by 8.8. In the LIE-style attack, all attackers submit the same coordinate-wise perturbation of the honest mean, bounded by z_{max} = 0.2275.

All three attacks are synchronous and white-box. The M = 10 malicious clients act in the same round and can observe the gradients of other clients. Sign reversal and the LIE-style attack share one forged vector per round, whereas label flipping does not, because every client trains on its own flipped data.

The LIE-style attack is a purpose-built variant of the original LIE attack [2]. It differs in three specified respects. The mean and standard deviation are estimated from honest updates under the white-box assumption, rather than from corrupted workers' own updates. The forged vector uses the lower-side perturbation shown in Table 1, reversing the direction of the original upper-side perturbation. The upper bound is obtained from a continuous inverse normal CDF, giving 0.2275, rather than the original tabulated grid value of 0.22. These departures preclude interpreting the results as a reproduction of the original attack.

Table 1. The three attack settings

Attack	Forged update
Targeted label flipping	all labels set to class 5, scaled by 8.85
Scaled sign reversal	$\theta = -\gamma \times \mu$, $\gamma = 8.8$
LIE-style	$\theta_j = \mu_j - z \times \sigma_j$, $z_{max} = 0.2275$

3.2. The two coded aggregation modes

FedNC and CHFL mix the updates of several clients before the server sees them [3, 4]. Let the mixing size m denote the number of clients in one group, with m = 1 meaning no mixing. Table 2 compares the two coded aggregation modes.

In Mode A (edge coding, aggregate after decoding), clients upload to a trusted edge server. The server forms a packet matrix $C = AP$, where A is the mixing matrix and P stacks the client updates. A is invertible, so the server recovers the individual updates by Gaussian elimination, then detects and aggregates. Mode A packs m updates into m packets, so it saves no uplink traffic, the 50%, 80% and 90% savings belong to Mode B only.

In Mode B (relay summation, aggregate after mixing), relays sum the updates before forwarding. Since the server only needs the sum, the aggregate is unchanged without detection, and each group's uplink drops from m vectors to 1. No coefficients are kept, so no individual update can be recovered, and detection can only act on the mixed result. The grouping layout describes how malicious clients are placed into packets, the spread layout scatters them across packets, while the consecutive layout packs them together.

Table 2. Comparison of the two coded aggregation modes

Dimension	Mode A	Mode B
Who mixes	edge server	relay
Server recovers individuals	yes	no
Where detection runs	after decoding	on the mixed result

3.3. Detector, metrics and research question

The detector is based on FLPD. For two clients, the cosine similarity is

$$cos(g_i, g_j) = \frac{g_i \bullet g_j}{\|g_i\| \times \|g_j\|} \tag{2}$$

The joint similarity φ_k averages this quantity over the n = 10 nearest neighbours:

$$\varphi_k = \frac{1}{n} \times \sum cos(g_i, g_j) \tag{3}$$

Each unit is scored by a cosine and a Euclidean criterion, and cross-round registries accumulate hits. Suspicious units are down-weighted, not removed, with tiers 1.0 / 0.4 / 0.15.

This detector is a simplified version of FLPD written for this study. The published article does not specify numerical values for the decision thresholds, clipping bound or down-weighting tiers. The present implementation also changes the decision procedure: every unit is scored with unconditional upper bounds based on the Interquartile Range (IQR), the baseline excludes a fixed fraction of high-similarity units, and Euclidean scores are normalized by the population mean norm. Gradient clipping is deliberately omitted. These are explicit implementation choices and departures from the published decision layer. Raw scores were not saved, so absolute detection coverage characterizes this simplified variant only. Neither the detector results nor the LIE-style results reproduce a published result.

Detection coverage is measured at two levels. Individual-level detection coverage (det) is the share of the M malicious clients that are down-weighted, averaged over the last 10 rounds. Packet-level det counts malicious clients inside down-weighted packets, divided by M, again over the last 10 rounds. The units differ, so the two measures cannot be compared across modes.

Registries only grow, so a flagged unit is never forgotten. A det of 100% therefore means all M attackers are down-weighted at that time, not that each round detected them on its own.

The experiments examine how detection coverage and model accuracy change under Mode A and Mode B as the mixing size m grows over 1, 2, 5 and 10.

4. Experiments and results

4.1. Experimental setup

All experiments are run in a single-machine PyTorch simulation. The dataset is Fashion-MNIST, with 10 classes and 60,000 training images [23]. A Dirichlet split with $\alpha = 0.5$ divides the data across $N = 100$ non-IID clients [24]. The model is a 6-layer CNN. Training uses a learning rate of 0.01, momentum 0.9 and 1 local epoch. The federation has $M = 10$ malicious clients with indices 0-9. These clients can forge their uploads and observe the gradients of other clients, so the setting is white-box. Table 3 lists the full parameter set. Unless stated otherwise, every headline number comes from $T = 50$ rounds and seed 0.

The notation rN denotes round N, and differences in accuracy are quoted in percentage points, written pt. Values reported with plus or minus are sample standard deviations over three seeds. In Figure 1, shaded bands span the available seed trajectories, undefended label flipping is shown as separate seed curves because its phase-dependent final values should not be averaged. In Figure 2, error bars span the minimum and maximum of the available three seed finals, with markers at seed 0. The flipping family is interpreted using peaks, last-10-round means and curve phase, an r50 reading alone is not an accuracy comparison. A reading of 10.00% is the trough of a collapse attractor, where the collapsed model predicts one class for every input.

Seed coverage is uneven. The individual-level runs, the sign-reversal Mode A runs and the sign-reversal Mode B spread runs have three seeds. The Mode A flipping runs, the consecutive block, the Mode B flipping runs and the additional defense variants have only seed 0. Within the LIE-style runs, only Mode B spread at $m = 10$ has three seeds, the other LIE-style configurations have seed 0 only. The LIE-style attack is a variant of the published attack [2]. None of the numbers reported here reproduces a published result.

An error in the present study must also be reported. In an early Mode B version, the individual-level aggregation step was missing its division by the mixing size m, so the summed update was m times too large. Even a run with no attack then collapsed to 10% at $m = 5$ and $m = 10$. This was a pure aggregation bug, and all 12 rows of the first, buggy run batch are void. The same error appeared once more in the undefended packet path of the LIE-style set, and a cross-model audit caught it. After the fix, a gating configuration confirmed the correction. The error is reported because it is the failure mode this paper studies. Such an error raises no exception and changes no logic in the code, it merely multiplies the server's effective step size by m. The same code therefore slides from apparently sound to fully broken as the mixing size grows. With no attack, the $m = 2$ run shows no cause for suspicion, while $m = 5$ and $m = 10$ both pin at 10.00%. It surfaces through coverage across m and through an independent audit, not through watching any single configuration.

Table 3. Finalized parameters

Category	Parameter	Value
Federation	Clients / malicious	$N = 100$ / $M = 10$ (indices 0-9)
Federation	Seed / rounds	0 (three seeds only for error bars) / $T = 50$
Data and model	Dataset / split	Fashion-MNIST / Dirichlet $\alpha = 0.5$
Data and model	Model / local epochs	6-layer CNN / 1
Training	Learning rate / momentum	0.01 / 0.9
Detector	Neighbours / exclusion	top 10 mean cosine / top 40%

Table 3. Continued

Detector	Down-weight tiers	1.0 / 0.4 / 0.15 (down-weighted, never removed)
Mixing	Mixing size m	2 / 5 / 10 (m = 1 is no mixing)
Mixing	Traffic saving	50% / 80% / 90% for m = 2 / 5 / 10
Mixing	Layout	spread / consecutive

4.2. Undefended dynamics and the individual-level FLPD baseline

The no-attack ceiling is 86.30% final accuracy (three seeds 86.09 plus or minus 0.30). The two attacks behave very differently. Scaled sign reversal gives a slow climb. It reaches 68.23% at r50 and is still climbing. Its net update amplitude is about 2%, which is not an accuracy rate: the honest step is scaled by (90 - 88)/100 = 0.02, where 90 is the number of honest clients and 88 the attackers' amplified mass (8.8 times 10). Targeted label flipping gives a collapse-attractor limit cycle. Accuracy stays flat at 10.00% with sparse repair spikes. In the collapsed state, the model predicts class 5 for every input, which is exactly 10% over 10 balanced classes. The final value is therefore phase-dependent, and averaging the three seeds is meaningless.

At the individual level, the detector removes the damage. Sign reversal ends at 85.37% (three seeds 85.18 plus or minus 0.26), about +17 pt above the 68.23% climbing transient. Flipping ends at 86.08% on a peak, with a last-10-round mean of 83.6%. A sawtooth oscillation still persists, but detection is perfect: det is 100% in 3/3 seeds, including the collapse-phase rounds of flipping. Figure 1 places the two dynamics together: the attack, the recovery, and the residual oscillation.

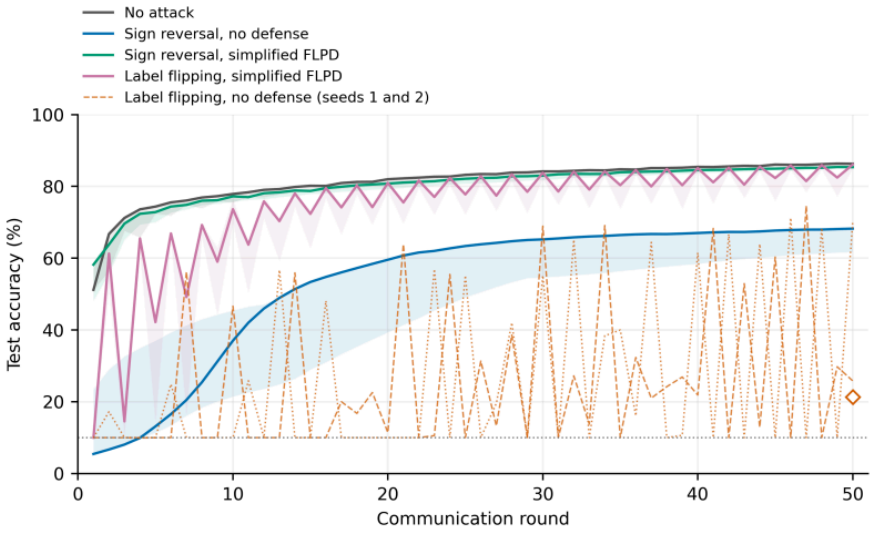


Figure 1. Accuracy over 50 rounds

Solid lines show seed-0 trajectories for the no-attack and defended runs and for undefended sign reversal, bands show the range of available seeds. Undefended label flipping is shown as separate seed-1 and seed-2 curves, without a mean. Its open diamond marks the archived seed-0 final (21.29%), for which no per-round trajectory is available. The dotted line marks the 10% collapse baseline. FLPD denotes the simplified variant.

4.3. Mode A: detection after decoding

In Mode A, the updates are first mixed into packets, and the server decodes them again with Gaussian elimination before it runs the detector. The round-trip error stays below 1e-4, so the recovered updates are the

individuals. Detection is unchanged. Figure 2(a) shows the flat lines: for m = 2, 5, and 10, the curves lie on top of each other. Sign reversal ends at 85.36%, 85.36%, and 85.41% against 85.37% at the individual level, within 0.04 pt. The flipping peaks are 86.01%, 85.99%, and 85.99% against 86.08%, within 0.09 pt, and the last-10-round means are about 81.6% against 83.6%. det is 100% in 6/6 configurations. The three-seed means agree to within 0.02 pt, with sample standard deviations up to 0.29 pt.

After decoding, the server receives the same type of per-client update as in uncoded FL, so the detector retains its individual-level inputs. This observed compatibility should be distinguished from the eavesdropping guarantees studied in FedNC [3]. Mode A adds online encoding and decoding and retains the same number of uplink vectors. The small phase differences in flipping runs are consistent with sensitivity to the random mixing matrix.

4.4. Mode B: detection on mixed packets

In Mode B, the server never decodes. Each packet is the plain sum of m updates, and the detector runs on packets. The initial expectation was that detection would fall monotonically with m. The measured behaviour does not follow that pattern.

Under the spread layout, the number of packets containing malicious updates is $\min(M, \text{number of packets})$. At m = 10 there are 10 packets, each containing 1 malicious and 9 honest updates, so none is clean. For the tested mixing sizes, consecutive grouping places malicious clients in packets containing only malicious members, at m = 10 they occupy a single packet. Figure 2(b) reports detection coverage with individual-level and packet-level units labeled separately.

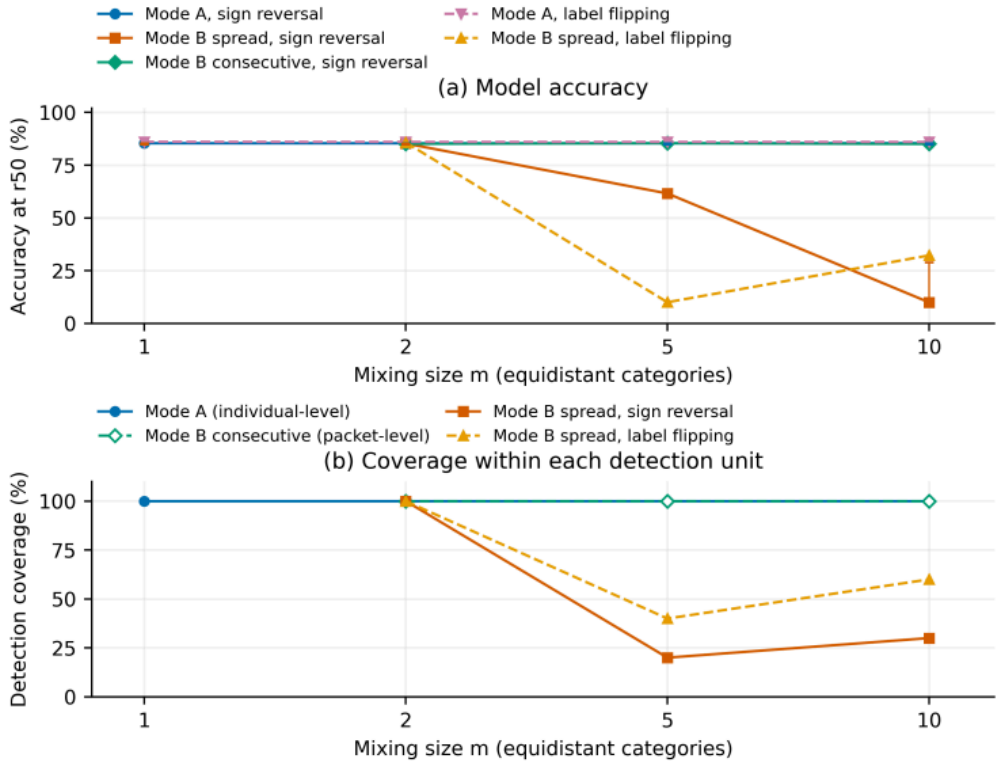


Figure 2. Accuracy and detection coverage against mixing size

Panel (a) gives archived r50 accuracy readings, bars span the three sign-reversal seed finals and markers show seed 0. Flipping endpoints are phase-dependent and should be read with the peaks and last-10-round

means in the text. Panel (b) reports cumulative down-weighting coverage averaged over the last 10 rounds: individual-level for Mode A and packet-level for Mode B. These units are not directly comparable. The $m = 1$ points are individual-level references, rather than additional Mode B runs.

Sign reversal is the strongest case. At $m = 5$, detection is pinned at 20%, which is almost no defense, a change of -6.6 pt against the 68.23% undefended run (final 61.59%). At $m = 10$, it is the strongest backfire in the paper. The accuracy curve climbs over r1-4, plateaus over r5-22, and decays from r23, it collapses through the 10% floor at r38 on seed 0. Detection is capped at 30% from r4. The defended model is worse than an undefended one: -58.2 pt against a matched undefended run at 68.23%, which used the same attack, m, layout, seed, and T. This is a paired comparison for seed 0 only. Seeds 1 and 2 have no undefended Mode-B spread run, so there the reference is the undefended individual level, which compares two different configurations instead of a matched pair of runs from the same one. The size of the backfire depends on the seed: the three seeds give defended finals of 10.00%, 29.03% and 21.89%. The backfire is real in all three, only its size changes. Flipping remains phase-dependent. At $m = 5$, the r50 reading of 10.02% is a collapse-attractor trough interrupted by repair spikes, at $m = 10$, packet-level det is 60% and r48-50 reads 37.97 / 21.31 / 32.22. For $m = 2$, 5 and 10, the last-10-round means are 80.97% / 33.01% / 28.81%, and the peaks within that same window are 85.90% / 71.82% / 68.34%. The $m = 2$ endpoint is near a peak, the $m = 5$ endpoint is a trough, and $m = 10$ remains in an irregular recovery phase. Accuracy comparisons therefore require matching curve phases or using the same averaging window.

In the evaluated configurations, full detection coverage accompanies healthy training, but lower coverage does not predict model accuracy monotonically. Within sign reversal, packet-level coverage of 20% and 30% accompanies accuracy differences of -6.6 and -58.2 pt relative to the stated undefended references. The flipping configurations instead have phase-dependent trajectories and must be interpreted using their peaks and last-10-round means. Across attack families, an LIE-style packet-level run remains at 86.03% with det of 23%, while the sign-reversal run with det of 30% collapses. These are different attack conditions, not a ranking of detector quality. Coverage describes which malicious contributions are down-weighted, it does not measure the damage imposed on honest contributors in the same packet.

The consecutive layout is the clean control, with the detector and mixing size fixed and only the placement of malicious clients changed. Packet-level det is 100% in all 50 of 50 rounds ($T = 50$) across the six sign-reversal and flipping configurations. At these tested mixing sizes, malicious packets contain no honest clients, so correctly flagging a malicious packet avoids penalizing honest contributors in that packet. This is conditional on correct flagging: det does not record false flags on pure-honest packets, and those false flags were not measured. Under the LIE-style attack at $m = 10$, packet-level coverage is 23% under spread and 100% under consecutive, a seed-0 difference of 77 pt, while accuracy differs by only 0.09 pt. Using the three-seed spread mean and the seed-0 consecutive value gives 66 pt, this is an asymmetric seed comparison. Table 4 gives the main spread results.

Table 4. Archived defended r50 readings and packet-level detection under spread

Attack (spread)	$m = 2$	$m = 5$	$m = 10$
Sign reversal, final acc	85.26%	61.59%	10.00%
Sign reversal, det	100%	20%	30%
Flipping, final acc	85.75%	10.02%	32.22%
Flipping, det	100%	40%	60%

4.5. The trade-off curve and the failure mechanism

Figure 3 shows client-level cosine similarity before packet mixing. Under sign reversal, the cosine between two malicious clients is about 1.000, while the cosine between two honest clients is about 0.014, or 0.025 with diagonal entries included. The identical malicious updates form a highly similar block. Under flipping, the malicious block is about 0.826 and the joint similarity oscillates between two bands. When attackers share an identical vector, each equally contaminated packet receives a common malicious component, packet-level scoring then lacks the separate malicious-client block visible in this figure. The heatmaps describe individual-update geometry and do not establish a general impossibility result for packet-level detection.

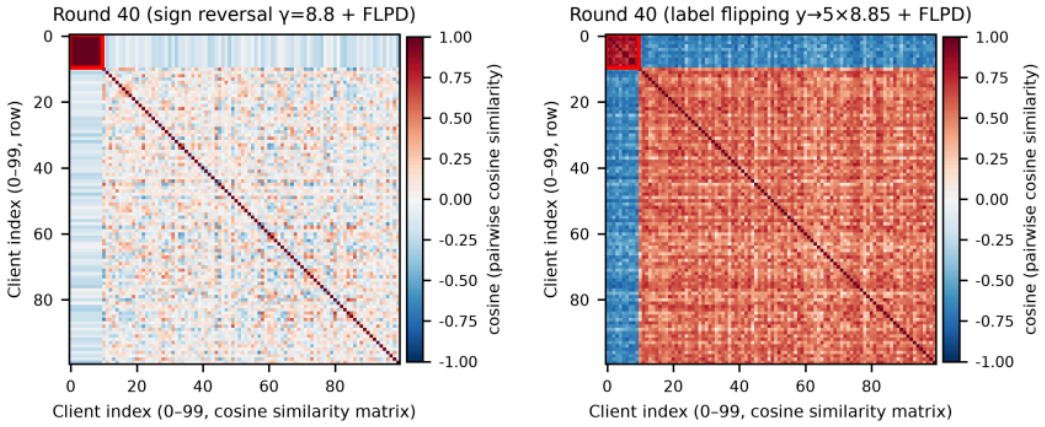


Figure 3. Client-update cosine similarity at round 40 before packet mixing. Left: scaled sign reversal. Right: targeted label flipping. The red box marks the malicious clients. Both runs use the simplified FLPD variant

The trade-off is therefore not a smooth curve of saved traffic against detection. Mode B's saved traffic may be taken out of detection capability, and how much is taken is decided by the layout, not monotonically by the mixing size.

5. Defense design and evaluation

5.1. Split-and-decode detection

In Mode B, a relay sends only a sum, which does not retain individual updates for detection. Recovering those updates requires enough independent coded packets and their mixing coefficients, as in the full-rank system used by Mode A [3, 6]. In the six evaluated sign-reversal and flipping configurations, decoding before detection gives individual-level det of 100%. This split-and-decode design is equivalent to Mode A rather than a new defense. Its traffic saving is 0%, in place of Mode B's 50%/80%/90%, so it provides a reference with individual-level detection and no uplink saving.

5.2. Group-based detection

Group-based detection retains Mode B's traffic saving. The relay sends each group sum, and the server is also given the membership of every packet. Detection acts on groups and downweights a suspicious group as a whole. Under the tested consecutive layout, packet-level det is 100%, but a deployment need not provide that layout. Under scaled sign reversal, the tested spread configuration produces defense backlash. A client-level variant, D3-lite, additionally uses grouping information constructed from known malicious indices. It is an oracle upper bound, not a deployable scheme. At the $T = 20$ gate, accuracy is 31.81% and client-level det is 16%, so the detection gate is missed. A sum does not identify which member supplied a malicious update. Under spread, no packet is clean once m is at least N/M, the 1 malicious plus 9 honest composition applies specifically at $m = 10$. Whole-group down-weighting therefore also penalizes honest members.

5.3. Comparison and choice

Table 5 compares the designs while identifying their detection units and evaluation horizons. Four changes to down-weighting were gated: removal of the lowest tier (D1a), removal of both down-weighting tiers (D1b), normalization by packet count (D2), and partner rotation with a client-level registry (D3-lite). Each was tested at $T = 20$ against preset accuracy and detection gates, both at least 30%. The first three use packet-level det, whereas D3-lite uses client-level det, their coverage values are not directly comparable. All four missed at least one gate. These are gating results, distinct from the main $T = 50$ evaluation. The matching $T = 50$ D3-lite run reaches 54.23% accuracy with client-level det of 0%. Avoiding the backfire is not the same as achieving a defense.

Table 5. The three designs, side by side, at $m = 10$

Design	Traffic saving	Detection	Accuracy
Mode B (spread, $m = 10$)	90%	packet-level det 30%	collapse
Split-and-decode	0%	individual-level det 100%	healthy
Group-based, client-level variant	90%	client-level det 16%	31.81%

The first two rows are $T = 50$ readings. The third is a $T = 20$ gate reading on the oracle D3-lite variant, not the deployable group-based design. Its matching $T = 50$ accuracy is 54.23% with client-level det of 0%. The rows use individual-level, packet-level and client-level definitions of det, which are not directly comparable.

The negative result concerns the four tested variants in the evaluated spread configuration at $m = 10 =$ N/M, with one common malicious vector per round and the simplified detector used here. It does not establish failure for every mixing size at or above N/M. It is not a general impossibility result, a theorem or a claim of structural necessity. Independently perturbed malicious updates may leave different signals across packets and remain untested. The evidence instead motivates treating detection resolution as a design variable alongside traffic saving and grouping control. Figure 4 shows the four $T = 20$ gating outcomes, with packet-level and client-level coverage labeled separately.

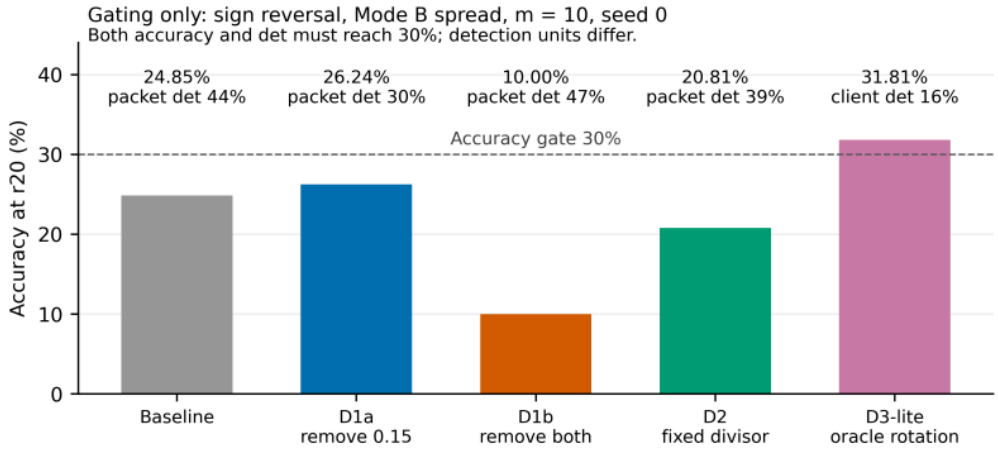


Figure 4. Gating results for the four tested variants and the packet-level baseline at $T = 20$ under sign reversal, Mode B spread and $m = 10$

Both accuracy and detection must reach 30%. D1a, D1b and D2 miss the accuracy gate, D3-lite is an oracle upper bound that misses the client-level detection gate. Detection annotations use different units and cannot be compared directly.

6. Conclusion

Mode A preserves individual-level detection in the six evaluated sign-reversal and flipping configurations, with det of 100%. Decoding restores individual updates before the detector operates. This compatibility does not reduce the number of uplink vectors.

Mode B changes the detection input from individual updates to group sums. In the LIE-style experiments at $m = 10$, packet-level coverage is 23% under spread and 100% under consecutive, while accuracy differs by only 0.09 pt. The individual-level reference has det of 100% under its separate definition and is not a directly comparable packet-level measure. Under scaled sign reversal at the same mixing size, the spread configuration can backfire and perform worse than its matched undefended run. The evidence concerns the interaction of attack, layout and defense in the tested settings.

In the evaluated spread configuration at $m = 10 = N/M$ with a common malicious vector per round, none of the four tested down-weighting variants passed both preset gates.

Coverage alone does not determine model health. The study has several limits. It uses a single-machine simulation without packet loss, latency or relay failure, the 50%/80%/90% traffic saving counts arriving vectors rather than bytes. Only three synchronous white-box attacks, one dataset and one $M : N$ ratio were evaluated. Sign reversal and the LIE-style attack share one malicious vector per round, label flipping instead trains separate updates on each client's flipped data. Independently perturbed malicious vectors were not tested, so the conclusions do not extend to that setting. Seed coverage is uneven, as specified in Section 4.1, the Mode A flipping, consecutive, Mode B flipping and additional defense runs have only seed 0. The detector is a simplified variant, and the LIE-style attack is purpose-built, so the results do not reproduce published algorithms. Reversible mixing in Mode A also has an information-theoretic privacy bound. Future work should study heterogeneous poison, adaptive attackers, and a real layered relay environment.

References

[1] McMahan, B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) (PMLR Vol. 54)* (pp. 1273–1282). PMLR.

[2] Baruch, G., Baruch, M., & Goldberg, Y. (2019). *A little is enough: Circumventing defenses for distributed learning.* arXiv. https://doi.org/10.48550/arXiv.1902.06156

[3] Shi, Y., Zhu, Z., Fan, P., Letaief, K. B., & Peng, C. (2024). FedNC: A secure and efficient federated learning method with network coding. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1–6). IEEE. https://doi.org/10.1109/WCNC57260.2024.10571245

[4] Gao, T., Lin, J., Li, C., Tan, C. W., & Gao, J. (2024). Federated learning meets network coding: Efficient coded hierarchical federated learning. In *Proceedings of the IEEE Information Theory Workshop (ITW)* (pp. 241–246). IEEE. https://doi.org/10.1109/ITW61385.2024.10806940

[5] Li, C., Xiao, M., & Skoglund, M. (2025). Coded robust aggregation for distributed learning under Byzantine attacks. *IEEE Transactions on Information Forensics and Security, 20*, 11636–11651. https://doi.org/10.1109/TIFS.2025.3624620

[6] Deng, J., Liu, S., & Li, C. (2024). Detecting diverse poisoning attacks in federated learning based on joint similarity. In *Proceedings of the 16th International Conference on Wireless Communications and Signal Processing (WCSP)* (pp. 133–138). IEEE. https://doi.org/10.1109/WCSP62071.2024.10827693

[7] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems (Vol. 30)*. Curran Associates.

[8] Yin, D., Chen, Y., Ramchandran, K., & Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning (ICML) (PMLR Vol. 80)* (pp. 5650–5659). PMLR.

[9] Cao, X., Fang, M., Liu, J., & Gong, N. Z. (2021). FLTrust: Byzantine-robust federated learning via trust bootstrapping. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Internet Society. https://doi.org/10.14722/ndss.2021.24434

[10] Fung, C., Yoon, C. J. M., & Beschastnikh, I. (2020). The limitations of federated learning in sybil settings. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)* (pp. 301–316). USENIX Association.

[11] Deng, J., Li, C., Zhang, N., Yang, J., & Gao, J. (2025). UDFed: A universal defense scheme for various poisoning attacks on federated learning. *IEEE Transactions on Information Forensics and Security, 20*, 10480–10494. https://doi.org/10.1109/TIFS.2025.3611126

[12] Wei, W., & Liu, L. (2024). Gradient coupling effect of poisoning attacks in federated learning. In *Proceedings of the 57th Hawaii International Conference on System Sciences (HICSS)* (pp. 7602–7611).

[13] Deng, J., & Li, C. (2025). A privacy-preserving network-coded federated learning system. In *Proceedings of the 2025 Asia Pacific Workshop on Data Science and Information Theory (APWDSIT)*. IEEE. https://doi.org/10.1109/APWDSIT64675.2025.11478553

[14] Zhang, Y., Gao, T., Li, C., & Tan, C. W. (2024). Coded federated learning for communication-efficient edge computing: A survey. *IEEE Open Journal of the Communications Society, 5*, 4098–4124. https://doi.org/10.1109/OJCOMS.2024.3423362

[15] Weng, S., Xiao, M., & Skoglund, M. (2026). *Coding-enforced resilient and secure aggregation for hierarchical federated learning.* arXiv. https://doi.org/10.48550/arXiv.2601.17995

[16] Jahani-Nezhad, T., Maddah-Ali, M. A., & Caire, G. (2025). ByzSecAgg: A Byzantine-resistant secure aggregation scheme for federated learning based on coded computing and vector commitment. *IEEE Transactions on Information Theory, 71*(8), 6410–6424. https://doi.org/10.1109/TIT.2025.3577019

[17] Liu, Y., Chen, C., Lyu, L., Jin, Y., & Chen, G. (2025). Exploit gradient skewness to circumvent Byzantine defenses for federated learning. *Proceedings of the AAAI Conference on Artificial Intelligence, 39*(18), 19024–19032. https://doi.org/10.1609/aaai.v39i18.34094

[18] Zhang, Z., Li, J., Yu, S., & Makaya, C. (2023). SAFELearning: Secure aggregation in federated learning with backdoor detectability. *IEEE Transactions on Information Forensics and Security, 18*, 3289–3304. https://doi.org/10.1109/TIFS.2023.3280032

[19] Xhemrishi, M., Östman, J., Wachter-Zeh, A., & Graell i Amat, A. (2025). FedGT: Identification of malicious clients in federated learning with secure aggregation. *IEEE Transactions on Information Forensics and Security, 20*, 2577–2592. https://doi.org/10.1109/TIFS.2025.3539964

[20] Xhemrishi, M., Graell i Amat, A., & Pejó, B. (2025). Detect & score: Privacy-preserving misbehavior detection and contribution evaluation in federated learning. In *Proceedings of the International Workshop on Secure and Efficient Federated Learning (FL-AsiaCCS)* (pp. 1–6). ACM. https://doi.org/10.1145/3709023.3737692

[21] Rathee, M., Zhang, Y., Corrigan-Gibbs, H., & Popa, R. A. (2024). Private analytics via streaming, sketching, and silently verifiable proofs. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)* (pp. 3072–3090). IEEE. https://doi.org/10.1109/SP54263.2024.00245

[22] Cheng, G., Yang, M., Zhao, X., Yu, S., Du, T., Wu, Y., Zhu, M., & Deng, S. (2026). LSHFed: Robust and communication-efficient federated learning with locally-sensitive hashing gradient mapping. *Proceedings of the AAAI Conference on Artificial Intelligence, 40*(25), 20490–20498. https://doi.org/10.1609/aaai.v40i25.39184

[23] Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms.* arXiv. https://doi.org/10.48550/arXiv.1708.07747

[24] Hsu, T. M. H., Qi, H., & Brown, M. (2019). *Measuring the effects of non-identical data distribution for federated visual classification.* arXiv. https://doi.org/10.48550/arXiv.1909.06335