Does Efficiency Leak? Auditing the Privacy of Compressed and Stale Updates in Federated Intrusion Detection for IoT

Chenglong Fu[1,✉], Xuening Xu[2], and John Du[3]

[1] University of North Carolina at Charlotte, United States
chenglong.fu@charlotte.edu
[2] Stevens Institute of Technology, United States
xuening0912@gmail.com
[3] University of Maryland, United States
johndu@terpmail.umd.edu

Abstract. Federated learning (FL) allows IoT gateways to train a shared intrusion detector while retaining raw traffic locally. Resource-constrained gateways often compress their updates and participate asynchronously, but the privacy effects of these efficiency mechanisms remain unclear. We evaluate an honest-but-curious server's ability to infer private traffic information from compressed, stale updates on three tabular intrusion-detection datasets. Batch aggregation limits record reconstruction, but gateway-level distributional information remains exposed: client-linkable membership inference reaches AUC 0.77 and remains statistically equivalent to inference from uncompressed updates. Compression attenuates attack-family attribute inference, which remains above chance at AUC 0.59–0.74, and reduces class-mixture leakage more substantially. Membership inference falls to chance when members and non-members come from the same gateway, indicating that the observed signal primarily reflects differences between gateway distributions. Error feedback increases the range of coordinates exposed across rounds, whereas staleness up to $\tau \leq 8$ has no measurable effect in the evaluated setting. Differential privacy reduces the measured leakage at a cost to detection utility; secure aggregation removes access to individual updates but can retain distributional leakage in the aggregate. Compression and asynchronous aggregation provide efficiency benefits but do not protect the remaining gateway-level information. Protecting it requires a separate privacy mechanism, with utility or deployment costs.

Keywords: Federated Learning · Intrusion Detection · IoT · Gradient Leakage · Communication Compression · Privacy

1 Introduction

Federated learning (FL) enables distributed Internet of Things (IoT) gateways to train a shared intrusion detector while retaining raw traffic locally [24,22,8].

✉ Corresponding author.

Packet traces and flow records can contain sensitive addresses, timestamps, and payload-derived features, making centralized collection difficult to justify or authorize. This motivates FL for intrusion detection, and recent benchmarks evaluate its detection performance on IoT traffic datasets [16,28,4,5].

Differences in gateway compute capacity and connectivity make synchronous training costly. Asynchronous and semi-asynchronous protocols accept updates from slow clients that trained on older model versions [40,29,39,30]. To reduce bandwidth and energy use, clients can combine top-k sparsification, low-bit quantization, and error feedback, which carries discarded update components into later rounds [2,3,23,31]. These mechanisms change both the content and timing of the updates available to the server, raising the question of how they affect privacy.

Gradient-inversion attacks can reconstruct training inputs from shared gradients [47,45,18], including under sparsification and quantization [43,14,44]. Learned inverters can also recover information under compression and noise settings that impede optimization-based attacks [38]. Most of this evidence concerns image data or fresh gradients. Federated intrusion detection instead exchanges model deltas computed from multiple local steps on tabular traffic. FedDef studies reconstruction from raw single-step gradients at batch size one and evaluates an input-perturbation defense [11]; TabLeak and SPEAR study tabular reconstruction from uncompressed updates [36,13]. These studies leave open how compression and staleness jointly affect leakage from multi-epoch intrusion-detection updates. Prior work on error feedback and asynchronous FL primarily evaluates training utility and convergence [35,42,40,29].

Our contributions are as follows.[4]

- We audit five privacy objectives in compressed, stale, multi-epoch federated intrusion-detection updates: record reconstruction, client-linkable membership, class-mixture inference, attack-family inference, and support-pattern inference.
- We show that limited record reconstruction coexists with gateway-level distributional leakage. Compression preserves client-linkable membership within the tested equivalence margin; attribute inference remains above chance, and class-mixture leakage is substantially reduced. Same-client controls reduce membership AUC to chance, identifying client linkability as the source of the signal.
- We isolate the effects of error feedback and staleness. Error feedback increases cumulative coordinate exposure, while memoryless top-k yields a more stable and more informative index pattern. Staleness up to $\tau \leq 8$ has no measurable effect in the evaluated setting.
- We quantify the costs of protecting the remaining leakage with differential privacy or secure aggregation. Sparse DP achieves comparable measured leakage and utility to dense DP at 80× lower uplink cost. DP's utility cost concentrates on non-dominant classes; secure aggregation removes

[4] Artifact: `https://anonymous.4open.science/r/fl-iot-paper-D98B/`.

individual-update access but can retain leakage where gateway mixtures are skewed.

2 Background and Related Work

2.1 Background

FL exchanges model updates computed from locally held data [24]. An honest-but-curious server follows the training protocol but may inspect these updates to infer private information. Attacks target several kinds of information: record reconstruction seeks individual training inputs [47,18]; membership inference tests whether a candidate record was used in training [27]; and property or distribution inference estimates characteristics of a client's dataset, such as its class mixture [25,34]. These objectives require separate evaluation. A low reconstruction score does not establish that the update conceals the client's distribution. Our audit measures these objectives for intrusion-detection updates that are sparsified, quantized, corrected by error feedback, and potentially stale.

2.2 Related Work

Gradient inversion under compression. Deep Leakage from Gradients demonstrated input and label recovery from shared gradients [47]. Subsequent work derived labels analytically [45], introduced cosine-based reconstruction objectives [18], and extended recovery to batches [13] and multi-epoch FedAvg deltas [41]. Reconstruction has also been demonstrated with as little as 0.1% of coordinates retained [43], at 80–90% sparsity [14], and under several forms of gradient obfuscation [44]. Learned inverters have been evaluated against compression and noise defenses [38,46], while OLIVE studies leakage from the top-k index set [20]. Broader evaluations show that attack success depends strongly on the training and system configuration [15,9,19]. This motivates testing the combined top-k+int8+error-feedback mechanism on stale, multi-epoch tabular updates.

Tabular intrusion detection and privacy defenses. TabLeak reconstructs mixed categorical and continuous features using softmax relaxation and pooled ensembling, evaluating both FedSGD and multi-epoch FedAvg [36]. SPEAR studies exact batch recovery [13]. We use these methods and metrics to guide the reconstruction evaluation. For network intrusion detection, FedDef evaluates raw single-step gradients at batch size one and proposes input perturbation [11]. Our audit extends this setting to compressed, stale, multi-epoch updates. Other federated intrusion-detection systems incorporate sparsification with differential privacy [12], secure multiparty masking [39], or homomorphic encryption [30]. These approaches provide complementary protection, with different effects on utility and deployment cost [32,7,1]. We compare compression, differential privacy, and secure aggregation under common leakage and detection metrics.

Staleness and error feedback. Asynchronous and buffered FL use age-dependent mixing weights to manage stale updates, with convergence as their

main evaluation target [40,29]. Temporal Gradient Inversion accumulates gradients across synchronous rounds [21]. Our staleness experiment instead fixes the training round and varies the model-version gap, separating staleness from training progress. Error feedback is usually evaluated for its ability to maintain utility during compressed training [35,42]. We measure how carrying the residual changes the coordinates exposed to the server and the information inferred from them.

3 System and Threat Model

3.1 System Model and Protocol

Figure 1 shows the system. We consider K IoT gateway sites, each collecting local network traffic $\mathcal{D}_k$ and training an IDS model. We model differences in compute capacity and uplink connectivity with three latency tiers: **Tier 1** (edge boxes, 30% of sites), with latency $\sim \mathcal{U}(3, 8)$ s; **Tier 2** (site gateways, 50%), with latency $\sim \mathcal{U}(20, 40)$ s; and **Tier 3** (remote collectors, 20%), with latency $\sim \mathcal{U}(80, 120)$ s. The tiers represent fast local links, moderate cellular/WiFi links, and constrained NB-IoT/cellular links, respectively.

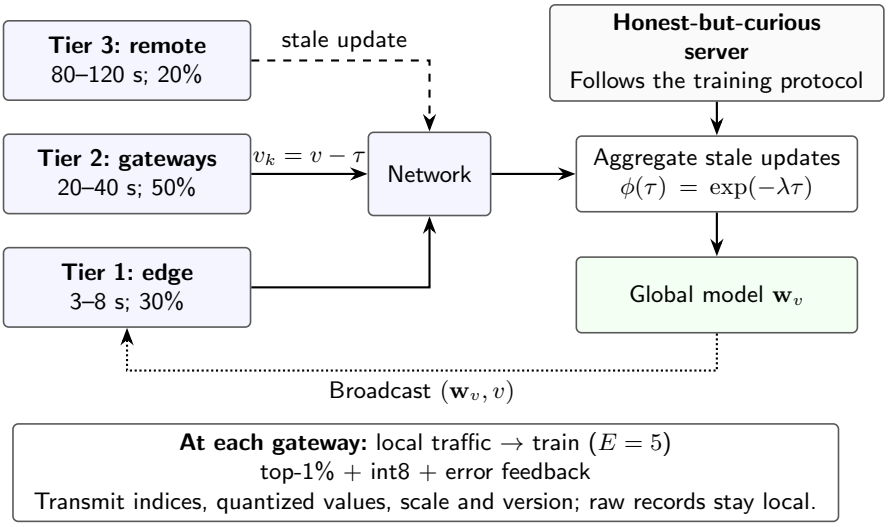


Fig. 1. Semi-asynchronous federated intrusion detection for IoT. Gateways across three latency tiers send compressed (top-1%, int8, error feedback), possibly stale (version $v - \tau$) updates to an honest-but-curious server, which aggregates with staleness weight $\phi(\tau) = e^{-\lambda\tau}$ and rebroadcasts. The server sees only the transmitted indices, int8 values, scale, and version, never the raw records.

Semi-asynchronous protocol. The server maintains a global model $\mathbf{w}$ with version counter v. The protocol operates in time windows of T_w seconds: the

server broadcasts $(\mathbf{w}_v, v)$ to selected gateways; gateway k pulls model version v_k, trains locally for E_k epochs on $\mathcal{D}_k$, compresses the model delta, and transmits it; the server collects all updates arriving within the window, then aggregates and increments v. Gateway k's staleness is $\tau_k = v_{\text{current}} - v_k \geq 0$, and updates with $\tau_k > \tau_{\max}$ are discarded.

Client-side compression. Each gateway compresses its model delta $\boldsymbol{\Delta}_k = \mathbf{w}_{k,\text{local}} - \mathbf{w}_{v_k}$ using error feedback (pseudocode in the supplementary material). The residual $\mathbf{r}_k$ accumulates the part of the delta that sparsification discards and re-injects it in later rounds; this is the mechanism we later audit as a leakage channel.

Server-side aggregation. The server decompresses updates, computes staleness, and aggregates:

$$\mathbf{w}_{v+1} = \mathbf{w}_v + \eta \sum_{k \in \mathcal{R}_v} \frac{n_k}{\sum_j n_j} \cdot \phi(\tau_k) \cdot \hat{\boldsymbol{\Delta}}_k \tag{1}$$

where $\phi(\tau) = \exp(-\lambda\tau)$ is the age weight and $\hat{\boldsymbol{\Delta}}_k$ is the decompressed delta. Staleness weighting is applied only at the server, after decompression, following the standard design of FedAsync [40]. When gateways perform different amounts of local work, each delta may be normalized by the number of local SGD steps, following Wang et al. [37].

Communication accounting. For a model with d parameters and sparsity ratio p, each gateway transmits pd uint32 indices ($4pd$ bytes), pd int8 values (pd bytes), a single per-message float32 scale s_k (4 bytes; the max-absolute transmitted value over 127, so dequantization is $s_k\mathbf{q}_k$), and a 4-byte version, for a total uplink of $5pd + 8$ bytes per gateway per round.

3.2 Threat Model

The adversary operates the central aggregation server and follows the training protocol while inspecting the updates it receives. For each participating gateway, this **honest-but-curious (HBC)** server observes $(\mathcal{I}_k, \mathbf{q}_k, s_k, v_k)$: the top-$k$ index set, int8 values, quantization scale, and client model version. It observes these fields before applying staleness weights. Raw traffic remains on the gateway. The server knows the model architecture, loss, feature schema, feature ranges, batch size, and number of local epochs, consistent with prior intrusion-detection leakage work [11]. Its inference targets include whether a candidate flow was used in training, which attack families a gateway has observed, and their proportions. Such information can reveal a site's threat exposure and help identify the organization associated with a gateway. The HBC setting is used in intrusion-detection leakage studies [11] and compression-defense evaluations [44]; active manipulation of the broadcast model is outside our main threat model.

Because the server sees an update before weighting it, server-side staleness weighting cannot conceal that update. The transmitted top-k indices also provide information beyond the quantized values [20]. We assume that the server can link updates from the same gateway across rounds and has a disjoint auxiliary corpus

with the same feature schema for training learned attack components. Linkability is relevant to the index and error-feedback evaluations. Client sampling reduces the number of available observations; secure aggregation restricts the server to a cohort aggregate. We evaluate secure aggregation [7] and differential privacy [1] separately as defenses. Two reference settings contextualize the HBC results: an *uncompressed single-step* update and an idealized *malicious per-example* ceiling. In the latter, the adversary receives a fresh gradient for each individual example, enabling closed-form inversion. This models a high-information reference motivated by active-server attacks [17,6]; we do not implement trap weights or other model-manipulation attacks.

4 Methodology

The audit examines five questions. **RQ1:** How accurately can an adversary reconstruct tabular traffic from compressed, stale, multi-epoch updates, using the client-mean record as the target for aggregated updates? **RQ2:** How do sparsification, int8 quantization, error feedback, batch aggregation, and local epochs affect reconstruction? **RQ3:** How much client-linkable membership, class-distribution, and attack-family information is exposed, and how does membership inference change under a same-client control? **RQ4:** How does error feedback change support coverage and index-based inference across rounds? **RQ5:** At a fixed training round, how does the model-version gap affect leakage when the adversary knows, or does not know, the client's reference model? We describe the datasets, attacks, controls, and metrics below; additional implementation details are in the Appendix.

4.1 Datasets and Federated Setup

We evaluate detection utility and communication cost on three IoT/IIoT datasets following Bilal et al. [5]: Edge-IIoTset [16], with 23 numeric packet-level features; CIC-IoT-2023 [28], with 46 flow-level features; and TII-SSRC-23 [4], with 79 CICFlowMeter features. Each dataset is subsampled to 80K records (64K train / 16K test), standardized, and mapped to five classes: Benign, DoS/DDoS, Recon, BruteForce, and Other. We use macro-F1 as the primary utility metric because the datasets are imbalanced. Training records are partitioned across $K = 100$ gateways using a Dirichlet($\alpha = 0.3$) split; Table 4 also evaluates other values of α.

The leakage audit uses Edge-IIoTset, CIC-IoT-2023, and UNSW-NB15 [26]. TII-SSRC-23 is excluded because every evaluated detector converges to majority-class prediction in our setup (Section 5.1), limiting its usefulness for comparing class-dependent leakage. UNSW-NB15 provides a general network-flow comparison with 37 features. We retain its encoded 11-category schema to preserve the attack-family labels. Edge-IIoTset has a more balanced class distribution, whereas CIC-IoT-2023 and UNSW-NB15 each have a dominant class. These differences allow us to examine leakage across packet- and flow-level features, class distributions, and traffic domains.

Table 1. Inference objectives and observed outcomes under the deployed compressed, stale channel. Reconstruction of the client mean is limited, while gateway-level inference remains possible.

Privacy unit	Question the server asks	Result
Record reconstruction	Can the server recover the client-mean flow record?	limited
Same-client membership	Which of a gateway's own records were used?	near chance
Client-linkability	Does a record match this gateway's distribution?	leaks
Gateway distribution	Which attack families, in what class mixture?	leaks (attenuated)
Update support	Which coordinates are exposed over rounds?	leaks

The detector is a four-layer multilayer perceptron (256-128-64-C) with LayerNorm and dropout, containing about 48–63K parameters depending on input width. Gateways use SGD with momentum 0.9, batch size 64, and $E = 5$ local epochs. The main training configuration runs for 100 rounds with 50% gateway sampling, $T_w = 15$ s, $\tau_{\max} = 8$, and staleness weight $\lambda = 0.3$. Compression combines top-1% sparsification, int8 quantization, and error feedback. Each transmitted delta aggregates training over the gateway's local shard: on average 640 records for the 64K-record IoT training sets and 64 records for the 6,400-record UNSW-NB15 training set. Utility is averaged over three seeds. Unless otherwise stated, leakage metrics use ten seeds and 95% confidence intervals; individual tables identify exceptions. Learned attacks use an auxiliary pool comprising 30% of gateways, disjoint from the victim gateways.

Capture schedules vary by experiment (Table 6). The distributional audit captures rounds 10, 50, and 90 of the 100-round run to sample early, middle, and late training. The defense comparison uses the final two rounds of a 30-round run. The staleness sweep fixes a training round ($R = 12$) and varies the version gap τ, with an additional transient-regime control. Reconstruction, learned-attacker, error-feedback, and sparsity experiments use the shorter horizons reported in the appendix. Comparisons between compressed and uncompressed channels use matched schedules within each experiment.

4.2 Audit Design

Table 1 summarizes the privacy units and the corresponding inference questions. We evaluate each separately because low reconstruction accuracy can coexist with inference about a gateway's traffic distribution.

Inference objectives. We evaluate five objectives. (1) *Record reconstruction* uses a learned inverter trained on same-schema auxiliary data [38], TabLeak's softmax relaxation and pooled ensembling for categorical fields [36], and the AGIC approximation for multi-epoch deltas [41]. Training the inverter on the observed channel allows it to adapt to compression, an important consideration when evaluating defenses [15]. (2) *Membership inference* scores a candidate record against a client's update, with several non-member pools used to distinguish record membership from client linkability. (3) *Class-distribution inference* estimates the update's class mixture from output-layer signs and magnitudes. (4) *Attribute inference* tests whether the gateway observed a particular attack

family. (5) *Index leakage* uses only the transmitted support set [20]. We evaluate the HBC server using the reference settings in Section 3.2.

Controls. A component ablation separates sparsification, int8 quantization, error feedback, and local epochs; a batch-size sweep evaluates the effect of aggregation on reconstruction (RQ2). The error-feedback control fixes the seed, data, batch order, and sparsity, and toggles residual carry-forward. We measure the cumulative union of transmitted coordinates, successive-support overlap, and index-based inference to relate support changes to leakage (RQ4). The staleness control fixes the evaluation round and client data while varying τ. An oracle-reference adversary knows the client's reference model; a realistic-reference adversary uses the server's current model. Comparing them distinguishes the effect of larger deltas from reference-model mismatch (RQ5).

Metrics. Membership and attribute inference are reported as ROC-AUC, with chance at 0.5. Class-distribution leakage is $\ell_{\mathrm{cd}} = \max(0, 1 - e/e_{\mathrm{aux}})$, where $e = \frac{1}{2}\|\hat{\pi} - \pi\|_1$ is the total-variation error between the inferred and true class priors, and e_{aux} is the error of predicting the auxiliary prior. Thus, 0 indicates no improvement over the prior and 1 indicates exact recovery. For aggregated updates, reconstruction targets the client's mean record; the malicious per-example reference targets individual records. We report $\ell = \max(0, 1 - \mathrm{RMSE}_{\mathrm{recon}}/\mathrm{RMSE}_{\mathrm{base}})$, with the auxiliary-set mean as the baseline prediction. The score is clamped at zero when reconstruction is worse than this baseline. On strongly imbalanced data, the baseline may already be close to the target mean, making the normalized score sensitive to small errors. We therefore interpret it alongside the batch-size control and the class balance of each dataset.

5 Experiments

We first evaluate detection utility and communication cost, then examine mean-record reconstruction (RQ1 and RQ2), client-linkable membership and distributional inference (RQ3), and the effects of error feedback and staleness (RQ4 and RQ5). We conclude by comparing leakage, utility, and bandwidth under differential privacy and secure aggregation.

5.1 Detection Utility and Communication Cost

Table 2 reports in-domain detection. Apart from the FedNova implementation discussed in the table note, macro-F1 differs by roughly 1 percentage point across methods on Edge-IIoTset and CIC-IoT-2023. The full compressed method stays within 0.2 to 1.1 points of FedAvg while transmitting about 80× fewer bytes per round: top-1%+int8 reduces the Edge-IIoTset uplink from 189 to 2.4 KiB, with similar reductions on the other datasets. On TII-SSRC-23, every method converges to majority-class prediction and obtains the same score. We exclude this dataset from the leakage audit because this behavior limits comparisons of class-dependent inference. The following experiments use Edge-IIoTset, CIC-IoT-2023, and UNSW-NB15.

Table 2. In-domain detection. Mean over 3 seeds, 100 rounds, avg. staleness $\bar{\tau} \approx 2.0$. Acc. = accuracy (%), F1 = macro-F1 (%). Ours^{-C} (wt.) = staleness-weighted aggregation without compression; Ours^{-W} (cmp.) = compression without staleness weighting; Ours (full) = both. Deployed compression cuts per-round uplink from 189/212/245 KiB (dense float32) to 2.4/2.7/3.1 KiB on Edge-IIoTset/CIC-IoT-2023/TII-SSRC-23, an 80× reduction on all three.

	Edge-IIoTset		CIC-IoT-2023		TII-SSRC-23	
Method	Acc.	F1	Acc.	F1	Acc.	F1
FedAvg	60.5	49.2	98.6	65.0	85.5	30.7
FedProx	60.5	49.2	98.6	65.2	85.5	30.7
FedNova†	56.6	43.6	96.1	40.5	85.5	30.7
FedAsync	60.2	49.1	98.6	64.8	85.5	30.7
FedBuff	60.5	49.2	98.6	64.9	85.5	30.7
Ours^{-C} (wt.)	60.3	**49.5**	98.6	65.1	85.5	30.7
Ours^{-W} (cmp.)	60.2	48.2	98.6	64.8	85.5	30.7
Ours (full)	60.3	49.0	98.5	63.9	85.5	30.7

†These are the original measurements from the implementation labelled FedNova in our code. It divides each client delta by the local step count before weighted averaging, without effective-step rescaling. This implementation limitation may affect its utility; the row is retained for completeness and should not be treated as a validated comparison with standard FedNova.

Per-class utility. A separate ten-seed evaluation without DP examines the macro-F1 gap by class (supplementary material). On Edge-IIoTset, compressed and uncompressed macro-F1 are 0.473 and 0.472, with per-class differences within the reported confidence intervals. On CIC-IoT-2023, compression reduces macro-F1 by about 0.6 percentage points. The gap occurs mainly in the minority Recon (0.541 vs. 0.560) and Benign (0.784 vs. 0.794) classes; the dominant DoS/DDoS class remains at 0.999.

5.2 Record Reconstruction (RQ1, RQ2)

Table 3 compares reconstruction across aggregated updates and the idealized per-example reference. The malicious per-example ceiling permits exact recovery on both datasets (leakage 1.000) through closed-form inversion of the first fully connected layer. For aggregated channels, we score recovery against the client-mean record. The deployed compressed, stale, multi-epoch update yields 0.081 ± 0.016 on Edge-IIoTset and 0.157 ± 0.068 on CIC-IoT-2023. The component ablation and batch-size sweep indicate that batch aggregation accounts for most of the reduction; increasing local epochs from one to five has little effect. Disabling error feedback raises Edge-IIoTset mean-record leakage to 0.297 ± 0.057. This is consistent with the more stable support of memoryless top-k, which repeatedly exposes similar coordinates (Fig. 2). The uncompressed single-step reference scores 0.000 on Edge-IIoTset because its batch-mean estimator does not improve on the auxiliary-mean baseline and the normalized score is clamped at zero.

Table 3. Reconstruction leakage (0 = baseline, 1 = exact) over ten seeds. Aggregated channels are scored against the client-mean record. Their scores remain below the malicious per-example reference, which recovers individual records exactly.

Channel	Edge-IIoTset	CIC-IoT-2023
Uncompressed, E=1	0.064±0.054	0.129±0.060
Uncompressed, E=5	0.070±0.047	0.175±0.071
Compressed, E=1, EF	0.093±0.086	0.135±0.068
Compressed, E=5, EF (deployed)	0.081±0.016	0.157±0.068
Compressed, E=5, no-EF	0.297±0.057	0.233±0.090
Bracket: uncompressed single-step (batch)	0.000±0.000	0.185±0.057
Bracket: malicious per-example ceiling	**1.000**±0.000	**1.000**±0.000

Batch size. We vary the batch size of an uncompressed single-step update and apply closed-form first-layer inversion, targeting the individual record at B=1 and the batch mean at $B > 1$. Edge-IIoTset leakage falls from 1.000 to 0.100, 0.002, and 0.000 at B=1, 4, 16, 64, respectively, supporting the role of batch aggregation in limiting reconstruction. CIC-IoT-2023 has a dominant class accounting for about 90% of traffic, making the batch mean close to the global mean and complicating interpretation of the normalized score. Its values are non-monotonic (1.000, 0.268, 0.257, 0.385 at the same batch sizes). The sweep therefore supports an empirical finding for the evaluated data and estimator, rather than a general privacy guarantee from larger batches.

5.3 Distributional Leakage (RQ3)

Membership, class-distribution, and attribute inference measure related aspects of the information an update reveals about a gateway's traffic (Table 4). Client-linkable membership reaches AUC 0.770 ± 0.027 on Edge-IIoTset and 0.660 ± 0.040 on CIC-IoT-2023 under compression. Both are statistically equivalent to their uncompressed counterparts within the paired-test margin δ=0.03 (Appendix). Here, non-members come from other gateways. Using a held-out-test pool gives similar results, whereas using records that the same gateway never trained on reduces AUC to chance (Table 5(b)). These controls indicate that the high AUC primarily reflects differences between gateway distributions.

Attack-family attribute inference remains above chance under compression, at AUC 0.74 and 0.62, but decreases significantly from the uncompressed values of 0.80 and 0.70. The paired test does not establish equivalence for this metric. Class-distribution leakage also decreases: on Edge-IIoTset it falls from 0.567 ± 0.047 to 0.319 ± 0.037, while the compressed score on CIC-IoT-2023 is 0.061 ± 0.040. The latter dataset's dominant class leaves less variation in gateway class mixtures for the attack to recover.

UNSW-NB15 provides a comparison on general network-flow data. Its cached evaluation subset contains 8,000 records, including 7,082 benign records (88.525%), and retains 11 encoded categories, including separate Backdoor and Backdoors labels. Membership AUC is similar with and without compression (0.645 vs. 0.642), while attribute AUC decreases from 0.704 to 0.594. Across the three datasets, compression reduces the class-mixture score by 44

Table 4. Distributional leakage (membership and attribute as AUC, chance 0.5; class-distribution as normalized leakage; ten seeds, 95% CIs). Top: deployed compressed channel vs. uncompressed at α=0.3. Bottom: robustness of the compressed channel across Dirichlet heterogeneity (α=0.3 is the deployed compressed row above).

Dataset	Setting	Membership	Attribute	Class-dist.
Edge-IIoTset	Compressed (α=0.3)	0.770±0.027	0.744±0.021	0.319±0.037
	Uncompressed	0.770±0.024	0.797±0.016	0.567±0.047
CIC-IoT-2023	Compressed (α=0.3)	0.660±0.040	0.616±0.020	0.061±0.040
	Uncompressed	0.651±0.041	0.699±0.033	0.181±0.058
UNSW-NB15	Compressed (α=0.3)	0.645±0.023	0.594±0.019	0.088±0.059
	Uncompressed	0.642±0.023	0.704±0.038	0.300±0.073
Edge-IIoTset	Compressed, α=0.1	0.844±0.021	0.799±0.017	0.431±0.028
	Compressed, α=1.0	0.666±0.029	0.742±0.035	0.099±0.050
CIC-IoT-2023	Compressed, α=0.1	0.748±0.026	0.719±0.027	0.301±0.035
	Compressed, α=1.0	0.566±0.018	0.569±0.032	0.000±0.000

to 71%: 0.567 → 0.319 on Edge-IIoTset, 0.181 → 0.061 on CIC-IoT-2023, and 0.300 → 0.088 on UNSW-NB15. These reductions are consistent with the sparsity sweep, in which class-distribution leakage increases with the number of retained output-layer coordinates (supplementary material). Compression thus affects the inference objectives differently: client-linkable membership changes little, while attribute and class-mixture inference are attenuated.

Robustness to heterogeneity. We vary the Dirichlet concentration across $\alpha = 0.1, 0.3, 1.0$ on both IoT datasets (Table 4, ten seeds). Membership and attribute AUC remain above chance throughout and generally increase as the partition becomes more skewed. The leakage therefore persists across the tested heterogeneity levels. UNSW-NB15 is evaluated only at $\alpha = 0.3$, so robustness to heterogeneity remains untested for that corpus.

Learned membership attacks. We also evaluate a shadow-trained MLP [33] using auxiliary-client (feature, member) pairs. Its features include cosine and magnitude-aware alignment, a support-restricted norm ratio, and the candidate's loss and logit margin. We compare per-class calibrated and uncalibrated variants with training-free cosine and calibrated-cosine attacks, using ROC-AUC and TPR at 5% FPR [10] (Table 5(a), five seeds). On compressed updates, the shadow-MLP obtains lower AUC than cosine: 0.750 vs. 0.786 on Edge-IIoTset and 0.654 vs. 0.689 on CIC-IoT-2023. Its uncompressed AUCs are 0.764 and 0.653, respectively. These tested attacks give similar compressed-versus-uncompressed comparisons, although they do not establish the performance of an optimal adversary. Calibrated and uncalibrated shadow-MLP results agree within confidence intervals for both datasets and both non-member pools. With same-client non-members, all evaluated attacks obtain AUC near 0.51 (Table 5(b)). We therefore find no measurable record-membership signal under this control; the above-chance results occur when the attack can exploit differences between gateway distributions.

Table 5. Membership inference on the deployed compressed channel. *(a)* Attack comparison (client-discrimination pool, five seeds): the shadow-trained MLP obtains lower ROC-AUC and TPR at 5% FPR than the training-free cosine attack. *(b)* Non-member pools (ten seeds): same-client non-members reduce AUC to chance for both attacks. Absolute AUCs differ slightly from Table 4 because this pipeline samples non-members without replacement and computes single-example gradients in evaluation mode.

	Edge-IIoTset		CIC-IoT-2023	
(a) Attack comparison: Attacker, AUC, TPR@5%FPR				
	AUC	TPR@5%	AUC	TPR@5%
Cosine (training-free)	**0.786**	0.224	**0.689**	0.200
Calibrated cosine	0.777	0.204	0.682	0.194
Shadow-MLP (learned)	0.750	0.152	0.654	0.167
(b) Non-member pool: AUC for cosine, shadow				
	cosine	shadow	cosine	shadow
Other-client (client-linkable)	0.776	0.765	0.702	0.673
Held-out test	0.776	0.769	0.713	0.656
Same-client held-out	0.507	0.506	0.515	0.514

5.4 Error Feedback and Staleness (RQ4, RQ5)

Error feedback (RQ4). Error feedback reduces round-to-round overlap in the transmitted support (Fig. 2). Jaccard overlap is 0.015 with error feedback versus 0.457 with memoryless top-k on Edge-IIoTset, and 0.011 versus 0.364 on CIC-IoT-2023. Carrying discarded components into later rounds increases cumulative coordinate coverage by about 2.6× over the evaluated horizon. The index pattern alone supports attack-family inference at AUC 0.59 to 0.68. Memoryless top-k yields higher index-based leakage, consistent with its more stable support and the reconstruction result in RQ1.

We next accumulate support over windows of R consecutive rounds (ten seeds). A binary-union representation produces little change: under error feedback, attribute AUC changes from 0.58 to 0.57 on Edge-IIoTset and from 0.60 to 0.62 on CIC-IoT-2023 as R increases from 1 to 16. Coverage of the output-layer coordinates used by this attack reaches 0.5 to 0.65 within a few rounds. Recording each coordinate's transmission frequency yields modest gains: AUC increases from 0.58 to 0.64 at R=4 on Edge-IIoTset and from 0.60 to 0.64 on CIC-IoT-2023 before plateauing. Memoryless top-k shows little benefit from accumulation. Thus, greater cumulative support coverage under error feedback translates into a modest improvement for the frequency-based attack, with little benefit for the binary-union attack.

Staleness (RQ5). We sweep the model-version gap τ for oracle-reference and realistic-reference adversaries (supplementary material). On Edge-IIoTset, membership AUC remains near 0.78 from τ=0 to τ=8, with the two adversaries within 0.01 of each other. On CIC-IoT-2023, AUC ranges from 0.64 to 0.67. A control before convergence produces the same qualitative result despite update norms growing by 1.2 to 1.9× with τ, depending on the dataset. Class-distribution and attribute metrics likewise show no measurable change. Within the evaluated range, staleness has little effect on these attacks, and knowledge of the exact client reference model provides no measurable advantage.

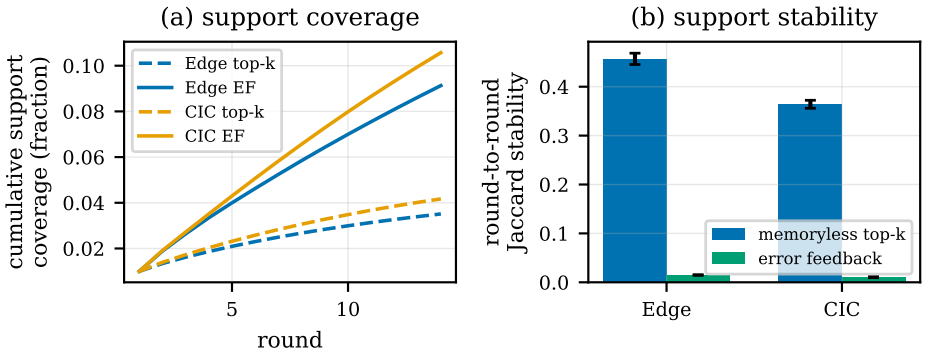


Fig. 2. Effect of error feedback on transmitted support (ten seeds). (a) Cumulative coverage, the fraction of parameters transmitted at least once, grows about 2.6× faster with error feedback than with memoryless top-k. (b) Round-to-round Jaccard overlap decreases under error feedback (0.015 vs. 0.46 on Edge-IIoTset).

5.5 Defenses and Frontier

Figure 3 compares compression, differential privacy, and secure aggregation over a 30-round defense experiment, using captures from its final two rounds. Because this schedule differs from the 100-round audit in Table 4, we compare results within the defense experiment. Compressed membership AUCs are 0.77, 0.67, and 0.62 on Edge-IIoTset, CIC-IoT-2023, and UNSW-NB15, close to the uncompressed values, with similar macro-F1. Dense DP clips updates and adds Gaussian noise to every coordinate. It reduces membership AUC toward chance and suppresses class-distribution leakage as σ increases, while macro-F1 falls from 0.48 to 0.26 on Edge-IIoTset and from 0.53 to 0.31 on CIC-IoT-2023. The corresponding client-level budgets, $\varepsilon \approx 132, 42, 17$, are large and provide weak formal bounds (Appendix). Most of the utility loss occurs in non-dominant classes: CIC-IoT-2023 Benign F1 decreases from 0.720 to 0.097, while the dominant DoS/DDoS class is nearly unaffected.

Dense DP transmits every coordinate. A sparse variant applies top-k+int8 after clipping and noising the dense update, making the transmitted indices and values post-processing of the same Gaussian mechanism. It reaches the same measured leakage floor with comparable or higher utility: at σ=0.5, sparse versus dense macro-F1 is 0.44 vs. 0.40 on Edge-IIoTset and 0.46 vs. 0.32 on CIC-IoT-2023. Re-sparsification reduces uplink cost by about 80× (supplementary material).

Our secure-aggregation evaluation assumes non-collusion and reveals only the per-round aggregate of 50 of 100 gateways. Its baseline uses held-out-test non-members, since every contributing gateway is represented in the aggregate. Replacing individual updates with the aggregate reduces membership AUC from 0.79 to 0.53 on Edge-IIoTset, from 0.69 to 0.59 on CIC-IoT-2023, and from 0.67 to 0.64 on UNSW-NB15. Secure aggregation removes direct observation of

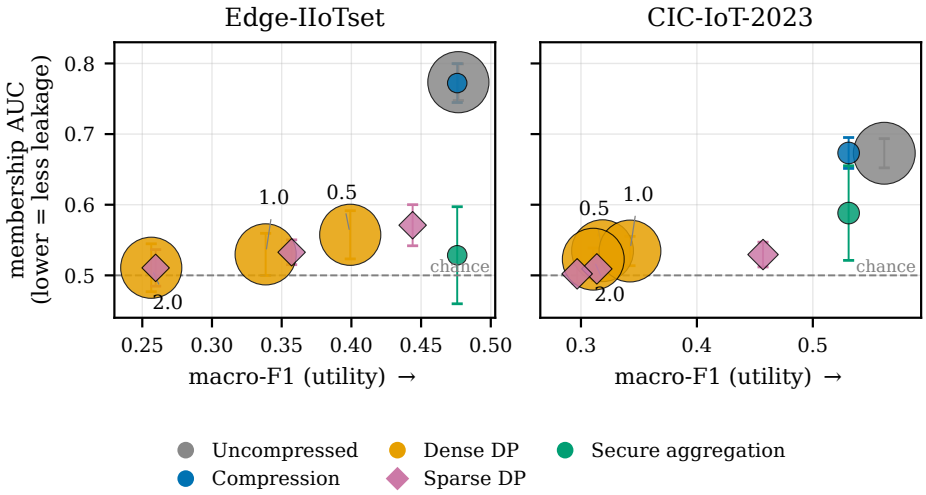


Fig. 3. Membership AUC versus macro-F1 on Edge-IIoTset and CIC-IoT-2023 (30-round defense experiment). Marker area represents uplink bytes; numeric labels show dense-DP noise multiplier σ. Compression (blue) gives similar leakage to uncompressed updates. Dense DP (orange) and re-sparsified DP (purple diamonds) reduce measured leakage as σ increases; re-sparsification lowers uplink cost by about 80×. Secure aggregation reduces AUC more on Edge-IIoTset. At $\sigma = 0.5, 1, 2$, client-level budgets are approximately $\varepsilon = 132, 42, 17$ ($\delta = 10^{-5}$, 30 rounds, full participation); these large budgets provide weak formal privacy bounds.

individual updates, but the remaining aggregate can still support distribution-based inference, with the measured reduction varying across datasets.

A sparsity sweep from top-0.1% to top-10% leaves membership AUC nearly unchanged on both IoT datasets, while class-distribution leakage increases with the number of retained output-layer coordinates (supplementary material). The UNSW-NB15 defense experiment shows similar trends: compressed and uncompressed membership AUCs are 0.62 and 0.63, within their confidence intervals, and DP reduces AUC to 0.53/0.51/0.50 at σ=0.5/1/2. Secure aggregation gives a smaller reduction, from 0.67 to 0.64.

Scope and limitations. The audit assumes an honest-but-curious server that follows the training protocol. Active model manipulation, malicious client selection, and adaptive probing could expose additional information and are outside this threat model; the per-example malicious ceiling is an idealized reference. All detectors are tabular MLPs, so transfer to CNN, LSTM, or GNN architectures remains untested. The 100-round distributional audit samples early, middle, and late training, and the staleness controls include transient and converged regimes. These controls test whether the findings depend on the selected checkpoint, but do not establish independence from detector quality. Edge-IIoTset macro-F1 remains modest, and separately optimized high-utility detectors have not been

audited. The conclusions are limited to the evaluated models, data, and training regimes.

6 Conclusion

Compression and asynchronous aggregation reduce the cost of federated IoT intrusion detection, but our audit shows that they do not protect the gateway-level information exposed to the server. Batch aggregation limits record reconstruction, while client-linkable membership, attack-family attributes, class mixtures, and support patterns continue to reveal information about gateway traffic. Compression leaves membership statistically equivalent to the uncompressed channel, attenuates attribute inference, and substantially reduces class-mixture leakage. The same-client control identifies client linkability as the source of the membership signal. Error feedback rotates the exposed coordinates and broadens cumulative coverage, while staleness up to $\tau \leq 8$ has no measurable effect. Protecting the remaining leakage therefore requires a separate privacy mechanism. Differential privacy incurs detection-utility costs, and re-sparsification retains the bandwidth saving; secure aggregation removes individual-update access but can retain an aggregate-level residual where gateway mixtures are skewed.

Disclosure of Interests The authors have no competing interests to declare that are relevant to the content of this article.

Appendix

Implementation details. The research artifact provides attack code, scripts for preparing the datasets, experiment configurations, compression pseudocode, attack descriptions, and supplementary staleness and sparsity tables, with pinned library versions. It is available at `https://anonymous.4open.science/r/fl-iot-paper-D98B/`. Across the checked BLAS and deep-learning-library versions, absolute AUCs and leakage scores vary by up to about 0.02, associated with top-k tie-breaking over repeated training rounds; paired comparisons retain the reported qualitative patterns. Learned attack components use Adam and an auxiliary pool of 30% of clients, disjoint from the victim clients. Table 6 specifies the capture schedules, followed by details of the privacy accountant and equivalence tests.

Differential privacy. DP-FedAvg clips each client update to L_2 norm $C = 0.5$ and adds Gaussian noise $\mathcal{N}(0, (\sigma C)^2)$ at $\sigma \in \{0.5, 1, 2\}$ over $T = 30$ rounds with full participation ($q = 1.0$). RDP composition of the per-round Gaussian mechanism, scanning integer orders for a valid upper bound, gives client-level $\varepsilon \approx 132, 42, 17$ at $\delta = 10^{-5}$. Full participation provides no subsampling amplification. These large budgets offer weak formal protection, especially at $\sigma = 0.5$; the experiment mainly quantifies measured leakage and utility as noise increases. Dense DP transmits the entire privatized update. Sparse DP clips and noises the dense update, then applies top-k+int8. Its transmitted support and values are

Table 6. Capture protocols for the audit. Round indices are zero-based as in the released code. Comparisons of compressed and uncompressed channels are paired within each experiment, not between rows. Configurations of auxiliary short-horizon runs are reported separately from the 100-round distributional audit.

Experiment	Horizon	Captured updates / evaluation
Detection utility	100	Final detector; no attack capture
Reconstruction (M2)	50	Rounds $5, 25, 45$
Distributional (M3), including α sweep	100	Rounds $10, 50, 90$
EF support dynamics (M4)	15	Rounds 1–14
Multi-round EF (M4b)	24	Rounds 1–23; sliding windows of $1, 2, 4, 8, 16$ captures
Staleness controls (M5)	Fixed v	Retrain against $\mathbf{w}_{v-\tau}$ while holding v and client data fixed; $\tau \leq 8$
Defense frontier (M6)	30	Rounds $28, 29$; DP utility after round 30
Learned attacker (M7)	15*	Round 10 under the released default configuration
Same-client control	15	Rounds $8, 14$
Sparsity sweep (M9)	15	Rounds $8, 14$

*The learned-attacker script also lists capture index 50, which is outside its default 15-round horizon. The archived result files do not retain command-line overrides, so this row specifies the reproducible released default rather than asserting an independently verified historical capture schedule. Its absolute AUCs should not be compared across capture pipelines.

post-processing of the same Gaussian mechanism and retain its privacy budget. The error-feedback residual contains only post-processed leftovers of already privatized releases; each new raw update is clipped before noising and composition. In these experiments, sparse DP reaches the same measured leakage floor (class-distribution score 0.00, membership approaching chance), at comparable or lower utility cost and 80× lower uplink. The attacks used in this DP comparison are non-adaptive to the added noise.

Paired equivalence test. We pair compressed and uncompressed channels across ten seeds, holding the data partition, batch order, and software environment fixed within each pair. Two one-sided t-tests (TOST) compare the per-seed differences with an equivalence margin $\delta = 0.03$ AUC; equivalence requires both tests to reject at the 5% level. This margin defines a tolerance of three AUC percentage points for comparing channels. It is small relative to the observed above-chance membership signal (0.14–0.27 AUC), but smaller changes may still matter at an operational false-positive rate. We therefore also report TPR at 5% FPR. The paired membership differences are +0.0001 on Edge-IIoTset, +0.009 on CIC-IoT-2023, and +0.003 on UNSW-NB15; all satisfy equivalence within δ with TOST $p < 0.001$. The CIC-IoT-2023 difference is also nonzero under a paired Wilcoxon signed-rank test ($p = 0.0137$), while the other two differences

are not significant. A small nonzero difference can satisfy equivalence within the specified margin. Attribute AUC decreases by 0.05 to 0.11, exceeding δ; a paired difference test finds a significant reduction, and equivalence is not established. Class-distribution leakage decreases more substantially (Section 5).

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: CCS (2016)
2. Aji, A.F., Heafield, K.: Sparse communication for distributed gradient descent. In: EMNLP (2017)
3. Alistarh, D., Grubic, D., Li, J., Tomioka, R., Vojnovic, M.: QSGD: Communication-efficient SGD via gradient quantization and encoding. In: NeurIPS (2017)
4. Alsaedi, M., Alzahrani, A., Alshehri, A.: TII-SSRC-23: An intrusion detection dataset for iot applications. In: IEEE Access (2023)
5. Bilal, M.A., Islam, I.U., Idrees, S., Qasim, M., Khan, M.J., Khan, J.: Dataset-centric evaluation of federated intrusion detection models in IoT networks. Scientific Reports **16**(2683) (2026). https://doi.org/10.1038/s41598-025-32567-w
6. Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A.S., Shumailov, I., Papernot, N.: When the curious abandon honesty: Federated learning is not private. In: 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P). pp. 175–199 (2023). https://doi.org/10.1109/EuroSP57164.2023.00020
7. Bonawitz, K., Ivanov, V., Kreuter, B., et al.: Practical secure aggregation for privacy-preserving machine learning. In: CCS (2017)
8. Briggs, C., Fan, Z., Andrzejewski, P.: A review of privacy-preserving federated learning for the Internet-of-Things. IEEE Access (2020)
9. Carletti, V., et al.: SoK: Gradient inversion attacks in federated learning. In: USENIX Security Symposium (2025)
10. Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., Tramèr, F.: Membership inference attacks from first principles. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 1897–1914 (2022). https://doi.org/10.1109/SP46214.2022.9833649
11. Chen, J., Zhao, Y., Li, Q., Feng, X., Xu, K.: FedDef: Defense against gradient leakage in federated learning-based network intrusion detection systems. IEEE Transactions on Information Forensics and Security **18**, 4561–4576 (2023). https://doi.org/10.1109/TIFS.2023.3297369
12. Chen, R., Chen, X., Zhao, J., Balfaqih, M.: Sparsified federated learning with differential privacy for intrusion detection in VANETs based on fisher information matrix. PLOS ONE **19**(4) (2024)
13. Dimitrov, D.I., Baader, M., Müller, M.N., Vechev, M.: SPEAR: Exact gradient inversion of batches in federated learning. In: NeurIPS (2024)
14. Ding, X., Liu, Z., You, X., Li, X., Vasilakos, A.V.: Improved gradient leakage attack against compressed gradients in federated learning. Neurocomputing **608**, 128349 (2024)
15. Du, J., Hu, J., Wang, Z., Sun, P., Gong, N.Z., Ren, K., Chen, C.: SoK: On gradient leakage in federated learning. In: USENIX Security Symposium (2025)
16. Ferrag, M.A., Friha, O., Hamouda, D., et al.: Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. IEEE Access **10** (2022)

17. Fowl, L., Geiping, J., Czaja, W., Goldblum, M., Goldstein, T.: Robbing the fed: Directly obtaining private data in federated learning with modified models. In: International Conference on Learning Representations (ICLR) (2022)
18. Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients: How easy is it to break privacy in federated learning? In: NeurIPS (2020)
19. Huang, Y., Gupta, S., Song, Z., Li, K., Arora, S.: Evaluating gradient inversion attacks and defenses in federated learning. In: NeurIPS (2021)
20. Kato, F., Cao, Y., Yoshikawa, M.: OLIVE: Oblivious federated learning on trusted execution environment against the risk of sparsification. Proceedings of the VLDB Endowment **16**(10) (2023)
21. Li, B., Gu, H., Chen, R., Li, J., Wu, C., Ruan, N., Si, X., Fan, L.: Temporal gradient inversion attacks with robust optimization. arXiv preprint arXiv:2306.07883 (2023)
22. Lim, W.Y.B., et al.: Federated learning in mobile edge networks: A comprehensive survey. IEEE Communications Surveys and Tutorials (2020)
23. Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W.J.: Deep gradient compression: Reducing the communication bandwidth for distributed training. In: ICLR (2018)
24. McMahan, B., Moore, E., Ramage, D., Hampson, S., Agüera y Arcas, B.: Communication-efficient learning of deep networks from decentralized data. In: AISTATS (2017)
25. Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 691–706 (2019). https://doi.org/10.1109/SP.2019.00029
26. Moustafa, N., Slay, J.: UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS). pp. 1–6 (2015). https://doi.org/10.1109/MilCIS.2015.7348942
27. Nasr, M., Shokri, R., Houmansadr, A.: 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 (2019). https://doi.org/10.1109/SP.2019.00065
28. Neto, E.C.P., Dadkhah, S., Ferreira, R., et al.: CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors **23** (2023)
29. Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M., Huba, D.: Federated learning with buffered asynchronous aggregation. In: AISTATS (2022)
30. Poorazad, S.K., Benzaïd, C., Taleb, T.: A novel buffered federated learning framework for privacy-driven anomaly detection in IIoT. In: IEEE GLOBECOM (2024)
31. Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., Pedarsani, R.: FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization. In: AISTATS (2020)
32. Ruzafa-Alcazar, P., Fernandez-Saura, P., Marmol-Campos, E., Gonzalez-Vidal, A., Hernandez-Ramos, J.L., Bernal-Bernabe, J., Skarmeta, A.F.: Intrusion detection based on privacy-preserving federated learning for the industrial IoT. IEEE Transactions on Industrial Informatics **19**(2), 1145–1154 (2023)
33. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 3–18 (2017). https://doi.org/10.1109/SP.2017.41
34. Suri, A., Evans, D.: Formalizing and estimating distribution inference risks. Proceedings on Privacy Enhancing Technologies **2022**(4), 528–551 (2022). https://doi.org/10.56553/popets-2022-0121

35. Valdeira, P., Xavier, J., Soares, C., Chi, Y.: Communication-efficient vertical federated learning via compressed error feedback. IEEE Transactions on Signal Processing **73** (2025). https://doi.org/10.1109/TSP.2025.3540655
36. Vero, M., Balunović, M., Dimitrov, D.I., Vechev, M.: TabLeak: Tabular data leakage in federated learning. In: ICML (2023)
37. Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V.: Tackling the objective inconsistency problem in heterogeneous federated optimization. In: NeurIPS (2020)
38. Wu, R., Chen, X., Guo, C., Weinberger, K.Q.: Learning to invert: Simple adaptive attacks for gradient inversion in federated learning. In: Conference on Uncertainty in Artificial Intelligence (UAI) (2023)
39. Xia, Z., Zhou, H., Hu, Z., Jiang, Q., Zhou, K.: Semi-asynchronous federated learning-based privacy-preserving intrusion detection for advanced metering infrastructure. International Journal of Critical Infrastructure Protection **49**, 100742 (2025). https://doi.org/10.1016/j.ijcip.2025.100742
40. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization. In: OPT Workshop at NeurIPS (2019), arXiv:1903.03934
41. Xu, J., Hong, C., Huang, J., Chen, L.Y., Decouchant, J.: AGIC: Approximate gradient inversion attack on federated learning. In: IEEE International Symposium on Reliable Distributed Systems (SRDS) (2022)
42. Xue, L., Hu, S., Zhao, R., Zhang, L.Y., Hu, S., Sun, L., Yao, D.: Revisiting gradient pruning: A dual realization for defending against gradient attacks. In: AAAI (2024)
43. Yang, H., Ge, M., Xiang, K., Li, J.: Using highly compressed gradients in federated learning for data reconstruction attacks. IEEE Transactions on Information Forensics and Security **18** (2023)
44. Yue, K., Jin, R., Wong, C.W., Baron, D., Dai, H.: Gradient obfuscation gives a false sense of security in federated learning. In: USENIX Security Symposium (2023)
45. Zhao, B., Mopuri, K.R., Bilen, H.: iDLG: Improved deep leakage from gradients. In: arXiv preprint arXiv:2001.02610 (2020)
46. Zhou, H., Qin, Z., Sun, P., Deng, X., Zou, Y., Wu, X.: Unifying gradient leakage attacks against privacy-protected federated learning in IoT networks. IEEE Internet of Things Journal (2026). https://doi.org/10.1109/JIOT.2025.3636944
47. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: NeurIPS (2019)