Article | Received 12 June 2026; Revised 15 September 2026; Accepted 29 October 2026; Published 10 November 2026
https://doi.org/10.55092/blockchain20260009

TrustFL-Chain: lightweight reputation-aware defense with tamper-evident auditing for non-IID federated learning

Ziyuan Zhong[1], Dun Li[2,*], Peiang Miao[1], Hongzhi Li[3] and Lijie Fan[4]

[1] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[2] Department of Industrial Engineering, Tsinghua University, Beijing, China
[3] College of Big Data and Artificial Intelligence, Chizhou University, Chizhou, China
[4] Jiashan Innovation Center, Yangtze Delta Region Institute of Tsinghua University, Jiashan, China

* Correspondence author; E-mail: lidun.2023@tsinghua.org.cn.

Highlights:

- Recoverable reputation and a hash-chain log support update-risk scoring.
- Five-seed results favor TrustFL-Chain under noise and inner-product attacks.
- Sign flipping and ALIE expose limits of the detector.

Abstract: Federated learning (FL) trains a shared model without collecting clients' raw data, but a server must still decide what to do with unreliable updates. Honest clients with different data can look unusual too. We study TrustFL-Chain, a lightweight prototype that scores updates, maintains recoverable client reputation, and writes aggregation decisions to a 256-bit Secure Hash Algorithm (SHA-256)-linked audit log. It does not require a trusted server dataset. Across 440 validated jobs, we test five Fashion-MNIST seeds, independent and identically distributed (IID) and non-IID partitions, six attacks, parameter sensitivity, unequal client sizes, and an exploratory Canadian Institute for Advanced Research (CIFAR)-10 supplement. With Dirichlet $\alpha = 0.5$ and 20% malicious clients, final-five-round accuracy exceeds Trimmed Mean by 6.18 percentage points under Gaussian noise. Under inner-product manipulation (IPM), TrustFL-Chain reaches 0.6457 accuracy against 0.3314 for the strongest observed baseline. Its 1.21-point mean lead over Cosine under sign flipping has a paired interval that crosses zero. Uniform averaging remains more accurate under the evaluated A Little Is Enough (ALIE)-style attack. In these experiments, reputation weighting helps most when manipulated updates leave a detectable signal.

Keywords: federated learning; poisoning defense; non-IID data; anomaly detection; reputation; tamper-evident audit log

1. Introduction

Federated learning (FL) allows data holders to train a shared model without sending their raw examples to a central site [1]. A 2021 survey describes both its promise and its persistent systems challenges [2].

Copyright©2026 by the authors. Published by ELSP. This work is licensed under Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium provided the original work is properly cited.

Collaborative training is especially appealing when moving data is costly, sensitive, or constrained by institutional rules [3]. Yet keeping data local does not make training robust [4]: the server may still receive updates that are mistaken, manipulated, or deliberately crafted.

Take an ordinary training round: a selected client trains locally and sends an update, which the server incorporates into the global model. If labels are corrupted, gradients reversed, noise injected, or a model delta amplified, later rounds may be affected. Attackers have used client-side control to manipulate FL models [5], including through model-replacement backdoors [6]. Practical constraints may limit some backdoors [7], but carefully shaped Byzantine updates can still bypass certain defenses [8].

Even honest training is uneven. Independent and identically distributed (IID) data provide a reference, but clients often have non-IID data with sharply different class proportions. Heterogeneity complicates communication and convergence [9]; local class imbalance can also degrade training [10]. Proximal objectives [11], control variates [12], adaptive server optimization [13], and normalization for different amounts of local work [14] each address part of the problem. Convergence analyses likewise show that heterogeneous clients change optimization dynamics [15]. For a poisoning defense, the immediate concern is that an honest client may produce an unusual update.

Against this backdrop, how can the server reduce a suspicious update's influence without judging a client permanently on the basis of one odd round? Several approaches have been proposed. Krum selects by distance [16]; coordinate-wise statistics [17] and Byzantine-resilient gradient rules [18] take other routes. Bulyan [19], generalized Byzantine-tolerant aggregation [20], and geometric-median methods [21] extend this line of work. Similarity-based defenses look for recurring geometric patterns [22], while trust bootstrapping may use a server dataset [23]. These defenses differ in the update geometry, client history, and trusted information they require.

An accuracy curve tells us whether training suffered, but little about the server decisions that led there. For an inspectable run, we also need the weights assigned to clients, the point at which their updates became suspect, and a way to notice later changes to the record. Distributed records have served coordination and auditing in blockchain-enabled FL [24]; surveys examine the associated trust, incentive, and decentralization questions [25]. Related work includes blockchain-FL at the edge [26], industrial Internet of Things (IoT) data sharing [27], and Biscotti's secure-learning protocol [28]. Blockchain has also been studied for digital twins [29], while cross-shard Byzantine fault tolerance addresses scalable ledger operation [30]. Sharding supports cross-domain sharing in zero-trust cloud-edge-end systems [31]. Smart-contract-inspired [32] and zero-knowledge-proof-inspired access control [33] pursue verifiability in constrained industrial systems. Those distributed guarantees are different from ours: the server keeps a local hash chain, whose retained hash allows later changes to be detected.

Figure 1 shows how TrustFL-Chain joins the learning and audit paths. Update features feed anomaly scoring, reputation updates, and weighted aggregation. Alongside those steps, the log stores update hashes, risk scores, reputation values, and aggregation decisions. A reader can inspect the recorded decisions after training.

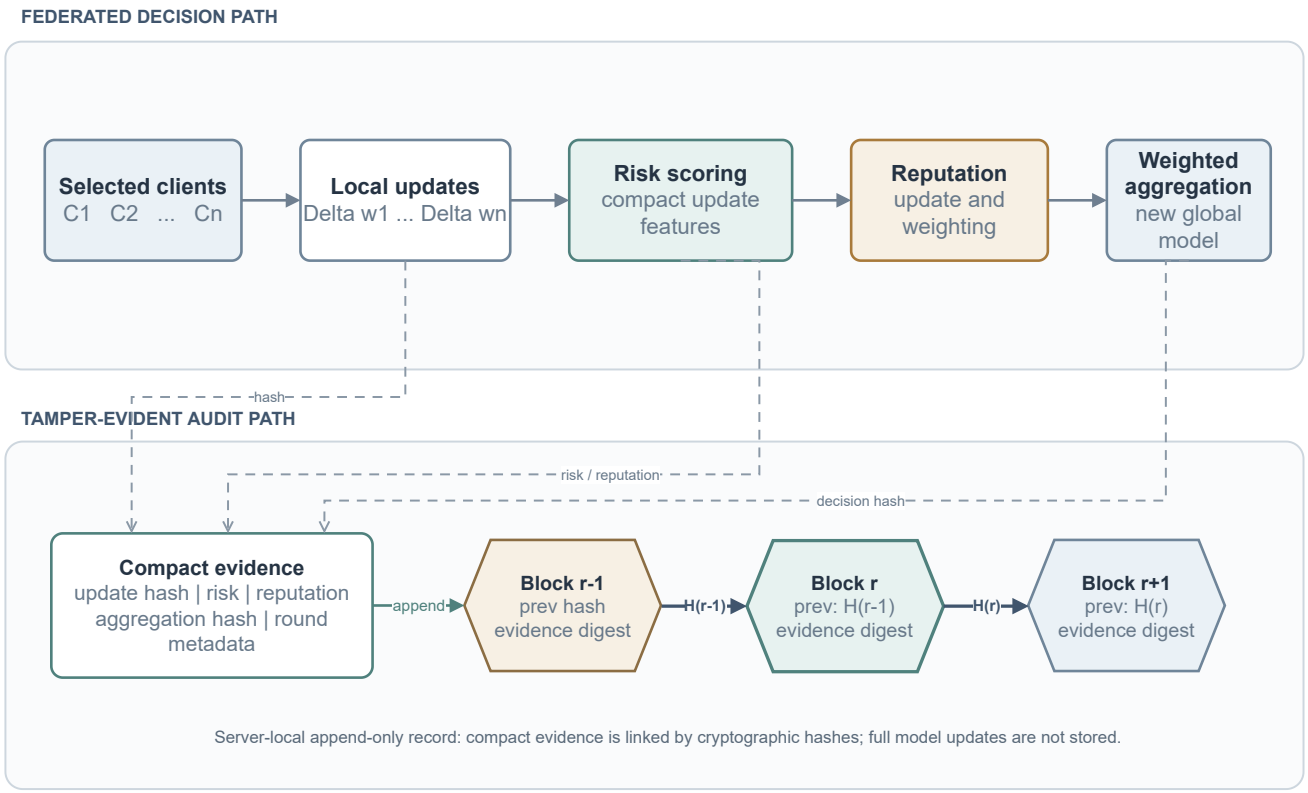

Figure 1. Conceptual role of the audit layer. Client updates are scored and weighted before aggregation, while compact evidence is committed to a server-local hash-linked log for traceability and reproducibility.

Our question is narrower than whether a ledger can secure FL end to end: can reputation-aware aggregation help under the attacks tested in a reproducible non-IID simulator, while leaving an inspectable record of the server's decisions? The server sees individual updates and keeps the hash-linked log locally; no decentralized party validates it.

Fashion-MNIST supplies the main multi-seed evidence across five parts of the evaluation: attacks, data partitions, baselines, ablations, and malicious-ratio sweeps. Canadian Institute for Advanced Research (CIFAR)-10 provides an additional optimization check. The attacks enter training in two ways: label flipping changes local training labels, while sign flipping, Gaussian noise, and model scaling directly change submitted updates.

Our contributions are fourfold:

- We implement TrustFL-Chain, a reproducible lightweight prototype for reputation-aware poisoned-update detection and compact hash-chain audit logging in non-IID FL.
- We evaluate robustness, detection, runtime, audit-log overhead, malicious-ratio sensitivity, component ablations, stronger attacks, and parameter sensitivity across 440 validated jobs.
- Five-seed results show the clearest gains under Gaussian noise and inner-product manipulation (IPM), while A Little Is Enough (ALIE) and sign flipping expose limits of the current detector.
- We analyze a targeted direction-filter variant for sign flipping and find that its local gain does not extend to the overall robustness profile.

2. Background and related work

2.1. Federated optimization, heterogeneity, and robust aggregation

FedAvg is a useful reference for three reasons: its averaging rule is simple, communication-efficient, and easy to reproduce. In the setting studied here, every selected client's update contributes to aggregation unless the server changes that rule. Different client training speeds can leave faster clients waiting; tier-based scheduling helps mitigate this delay [34]. A communication-efficient federated learning method with periodic averaging and quantization (FedPAQ) instead reduces communication costs [35].

These methods address the cost and pace of training. Robust aggregation addresses a different question: which updates should shape the next global model? It limits the effect of extreme or inconsistent updates, often using only the current cohort. Such methods are useful baselines, but they do not ordinarily retain client-level trust history or a record of why an update received less weight. That distinction matters when an honest client's data produce an unusual update.

2.2. Poisoning, backdoors, and trust-aware defenses

Model replacement is only one route available to an attacker. Local updates can be optimized against Byzantine-robust aggregation [36] or redirected through inner-product attacks [37]. Attackers may also corrupt training labels [38] or exploit rare subpopulations for backdoors [39]. Defenses take different forms: robust learning rates control update scale [40]; backdoor detection via feedback-based federated learning (BAFFLE) uses feedback [41]; and DeepSight inspects model behavior [42]. Attack-defense optimization has also been studied directly [43], while certification offers a different form of backdoor assurance [44]. The FLAME backdoor-defense framework combines clustering, clipping, and noise [45]. Our evaluation includes History-Cosine as a lightweight historical-similarity comparator. TrustFL-Chain maintains client reputation without a trusted root dataset.

2.3. Blockchain, auditability, and industrial trust

Auditing in FL starts with two questions: what can the server see, and what does it record? The guarantees of a defense depend on its threat model and deployment conditions [46]; from an attacker's perspective, those conditions also determine what information can be exploited [47]. We assume that the server can inspect each client's submitted update to assess its risk. This visibility has a privacy cost: gradients can disclose training information [48], and white-box access can support further inference [49]. Client-level differential privacy would change what the server can observe [50].

Blockchain and FL have been combined in several application settings. Basak *et al.* studied collaborative training in a medical cyber-physical system [51]. Other work addresses lightweight security in resource-constrained medical systems [52], energy use in Internet of Medical Devices deployments [53], and privacy-preserving FL with blockchain auditing in consumer electronics [54]. These systems have different goals; in each case, the meaning of auditability depends on what is recorded and whom the system trusts.

Here, the record concerns the aggregation server's handling of updates. Robust aggregators usually decide from the current cohort; trust-aware methods also use client history, and some require trusted server data. TrustFL-Chain combines update-risk scoring, recoverable reputation, and aggregation

decision records in a non-IID poisoning evaluation. Table 1 compares aggregation, history, root-data, and audit assumptions across representative methods.

Table 1. Positioning against representative defenses. "History" denotes persistent client state, "root data" a trusted server dataset, and "audit" a record produced by the method itself.

Method	Aggregation	Inspection	History	Root data	Audit	Primary distinction
Krum [16]	Robust	Distance	No	No	No	Selects a central update
Trimmed Mean [17]	Robust	Coordinate	No	No	No	Removes coordinate extremes
FoolsGold [22]	Weighted	Similarity	Yes	No	No	Historical sybil similarity
FLTrust [23]	Weighted	Direction	No	Yes	No	Trust from a server update
FLAME [45]	Robust	Cluster/clip	No	No	No	Backdoor clustering and noise
DeepSight [42]	Filtering	Behavior	No	No	No	Deep model inspection
Blockchain-assisted FL [54]	Varies	Varies	Varies	Varies	Distributed	Privacy and ledger integrity
TrustFL-Chain	Weighted	Five signals	Yes	No	Local	Recoverable trust plus audit log

3. Threat model and method

3.1. Threat model

We consider a synchronous FL setting with one aggregation server and $N = 20$ clients. At each round, the server samples $K = 10$ clients. A fixed fraction of clients is malicious. Malicious clients may alter local labels or manipulate submitted updates. We evaluate four attacks:

- Label flipping: malicious clients train on corrupted labels.
- Sign flipping: malicious submitted updates are multiplied by -1.
- Gaussian noise: malicious submitted updates receive additive noise.
- Model scaling: malicious updates are multiplied by a large scale factor.

We additionally evaluate two convergence attacks. The ALIE-style attack replaces every active malicious update by $\mu_m - z\sigma_m$ with $z = 1.5$, using the coordinate-wise mean and standard deviation of active malicious updates. IPM submits $-\varepsilon\bar{\Delta}_h$ with $\varepsilon = 5$ and active honest mean $\bar{\Delta}_h$. Single-seed strength checks use $z \in \{0.5, 1.0\}$ for ALIE and $\varepsilon \in \{1, 10\}$ for IPM.

The server follows the prescribed defense, inspects submitted updates, and maintains the audit record. Attackers control some clients' local training or submitted updates, but cannot alter the server program or log. Per-client risk scoring requires the server to read individual updates, so it cannot be combined directly with standard secure aggregation, which reveals only the aggregate [55]. The tested attacks do not adjust to the detector's internal features. Backdoor triggers, deployment networking, and consensus-layer attacks are not simulated. Within this setting, we measure model accuracy, detection, false positives under non-IID data, and audit overhead.

3.2. Pipeline overview

Figure 2 summarizes the implemented pipeline. Each selected client trains locally and submits an update. TrustFL-Chain extracts update features, estimates risk, updates reputation, aggregates with trust-aware weights, and records compact evidence in a hash-chain audit log.

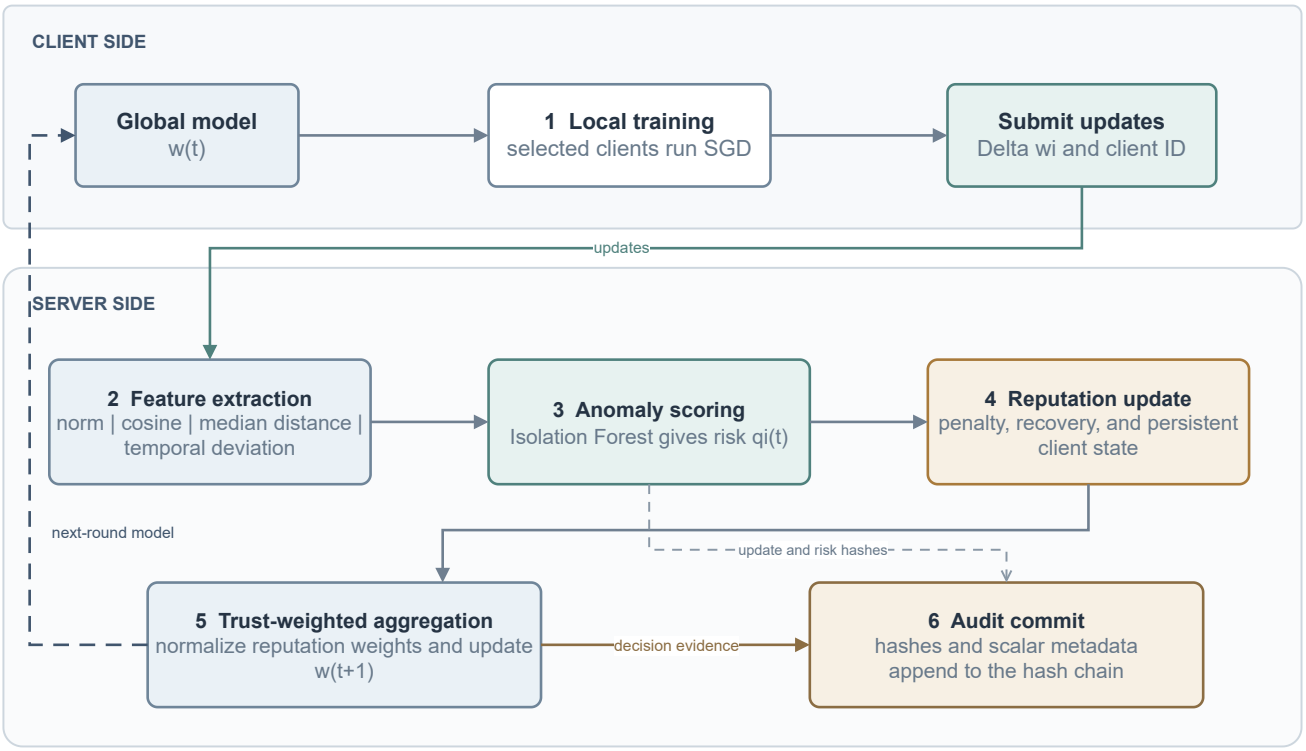

Figure 2. Implemented TrustFL-Chain pipeline. In each round, the server extracts compact features from submitted updates, estimates update risk, updates client reputation, performs reputation-aware aggregation, and records hashes plus scalar metadata in an append-only audit chain. Full model updates are not stored in the log.

3.3. Update features

For a selected client i at round t, let $g_i^t = \mathrm{vec}(\Delta_i^t)$ and let ε_0 be a small numerical constant. The prototype computes

$$n_i^t = \|g_i^t\|_2, \tag{1}$$

$$c_i^t = \frac{\langle g_i^t, \bar{g}^t \rangle}{\|g_i^t\|_2 \|\bar{g}^t\|_2 + \varepsilon_0}, \tag{2}$$

$$d_i^t = \|g_i^t - m^t\|_2, \tag{3}$$

$$u_i^t = \begin{cases} 0, & \text{first participation}, \\ \|g_i^t - g_i^{\tau_i}\|_2, & \text{otherwise}, \end{cases} \tag{4}$$

where $\bar{g}^t$ is the mean selected update, m^t is the coordinate-wise median, and $\tau_i < t$ is client i's most recent participation. A client absent for several rounds is compared with its last observed update, not with the preceding global round. The stored row is $x_i^t = [n_i^t, c_i^t, d_i^t, 0, u_i^t, r_i^{t-1}]$. The fourth column is reserved for validation-loss change and is zero in all reported runs because no trusted validation set is assumed; the other five columns vary.

3.4. Risk scoring, reputation, and aggregation

The feature vector is standardized within each round and passed to a lightweight anomaly detector. The default detector is Isolation Forest with contamination 0.2, matching the main simulated malicious ratio.

Alternatives such as local outlier factor (LOF), one-class support vector machine (SVM), and cosine-only scoring are used for ablations. The detector outputs a risk score $q_i^t \in [0,1]$. Contamination values 0.1 and 0.3 test moderate mismatch to the default prior.

Each client starts with reputation $r_i^0 = 1$. A risk score at or above the threshold reduces reputation in proportion to risk; a lower score allows gradual recovery:

$$r_i^{t+1} = \begin{cases} \max(r_{\min}, r_i^t(1-\lambda q_i^t)), & q_i^t \geq 0.5, \\ \min(1, r_i^t + \rho(1-r_i^t)), & q_i^t < 0.5. \end{cases} \tag{5}$$

The defaults are $r_{\min} = 0.05$, penalty $\lambda = 0.35$, recovery $\rho = 0.05$, and detection threshold 0.5. Unselected clients keep their previous reputation. Recovery prevents one anomalous round from permanently determining a client's weight: under non-IID data, an honest client may occasionally produce an update unlike those of the current cohort. Aggregation uses the updated reputation as its weight:

$$\Delta^t = \frac{\sum_{i \in S_t} r_i^{t+1} \Delta_i^t}{\sum_{i \in S_t} r_i^{t+1}}. \tag{6}$$

3.5. Audit log and direction-filter variant

Alongside aggregation, the audit log records the round number, selected clients, update hashes, risk scores, reputation values, aggregation hash, timestamp, and previous block hash. Full updates are not stored. The 256-bit Secure Hash Algorithm (SHA-256) links each serialized block to its predecessor, allowing consistency checks against a retained block hash. A server that controls the entire unanchored file could rewrite it, so the guarantee is local tamper evidence.

We also tested a stricter direction filter after the sign-flip results. It keeps the most direction-consistent updates before aggregation. The focused sign-flip rerun improves, while Gaussian-noise and model-scaling results worsen. We report the variant separately and retain the submitted aggregation rule.

4. Experimental design

Four research questions (RQs) guide the evaluation:

- RQ1: Does TrustFL-Chain improve final accuracy under common poisoning attacks?
- RQ2: Does the method produce useful malicious-client detection signals, not just robust aggregation?
- RQ3: Which components matter, and how does the method respond to malicious ratio, detector settings, reputation recovery, and unequal client sizes?
- RQ4: What are the runtime and audit-storage costs, and what can be concluded from the harder dataset?

Table 2 gives the evaluation matrix. Fashion-MNIST supports the wider multi-seed comparison; CIFAR-10 adds a smaller check under the available compute budget. All runs use a single-machine PyTorch simulator.

Table 2. Final evaluation matrix. Fashion-MNIST is the main multi-seed evidence. CIFAR-10 is supplemental.

Item	Setting
Main dataset	Fashion-MNIST
Supplemental dataset	CIFAR-10
Clients	20 total, 10 sampled per round
Main rounds	30 for Fashion-MNIST
CIFAR-10 rounds	20
Main seeds	42, 43, 44, 45, 46
Data splits	IID, Dirichlet $\alpha = 0.5$, $\alpha = 0.1$
Main malicious ratio	0.2
Attacks	label/sign flip, noise, scaling, ALIE, IPM
Main baselines	FedAvg, Trimmed Mean, Cosine, TrustFL-Chain
Additional baseline	History-Cosine for core direct-update attacks
Ablations	no reputation, no temporal, cosine-only, no audit log
Validated jobs	440 completed, 0 missing cells

Local training uses stochastic gradient descent (SGD). The remaining implementation and reporting choices, including seed-level standard deviations (SD), are collected in Table 3.

Table 3. Implementation and reporting details. These settings are fixed unless otherwise noted.

Item	Value
Framework	PyTorch simulator
Optimizer	Local SGD
Batch size	64
Train/test cap	10,000/2000 examples per run
Client sampling	10 of 20 clients per round
Primary seeds	42–46
TrustFL detector	Isolation Forest over update features
TrustFL aggregation	Reputation-weighted averaging
Reputation policy	Soft penalty and gradual recovery
Audit-log content	Hashes, scores, reputation, aggregation hash
Main reporting	Mean, SD, and seed-level points where applicable
Statistical tests	Paired intervals/tests, Holm adjustment, $n = 5$

The main Fashion-MNIST matrix compares attacks, data splits, and baselines across five seeds. A separate sweep examines model scaling at 10%, 20%, 30%, and 40% malicious clients. Ablations remove reputation, temporal history, the multi-feature detector, or the audit log. Further controls vary detector contamination, penalty, recovery, initial reputation, and threshold. They also compare uniform with sample-weighted aggregation and include clean CIFAR-10 runs.

We report final and best accuracy, precision, recall, F1 score (the harmonic mean of precision and recall), false positive rate (FPR), runtime per round, audit-log bytes, and reputation trajectories. To characterize late-stage performance, final-five-round accuracy averages rounds 26–30 for Fashion-MNIST

and 16–20 for CIFAR-10. Each selected client-round is an observation for detection; client-level detection uses a client's maximum risk over the run. We also report average precision (AP), delay measured in malicious-client participations, and the number of benign participations required to regain 90% of reputation lost after a false alarm. Recovery not observed by the end of a run is right-censored. Paired Fashion-MNIST comparisons use the same five seeds and report mean differences, 95% Student-t confidence intervals (CIs), and Holm-adjusted p values within prespecified families. With five pairs, the smallest attainable two-sided exact sign-permutation p value is 0.0625.

5. Results and analysis

5.1. RQ1: robustness under poisoning

Figure 3 shows the main accuracy comparison. Gaussian noise yields the clearest gain: final-five-round accuracy is 0.7672 ± 0.0207 for TrustFL-Chain and 0.7054 ± 0.0206 for Trimmed Mean. Under model scaling, the corresponding means are 0.7066 ± 0.0115 and 0.7000 ± 0.0155.

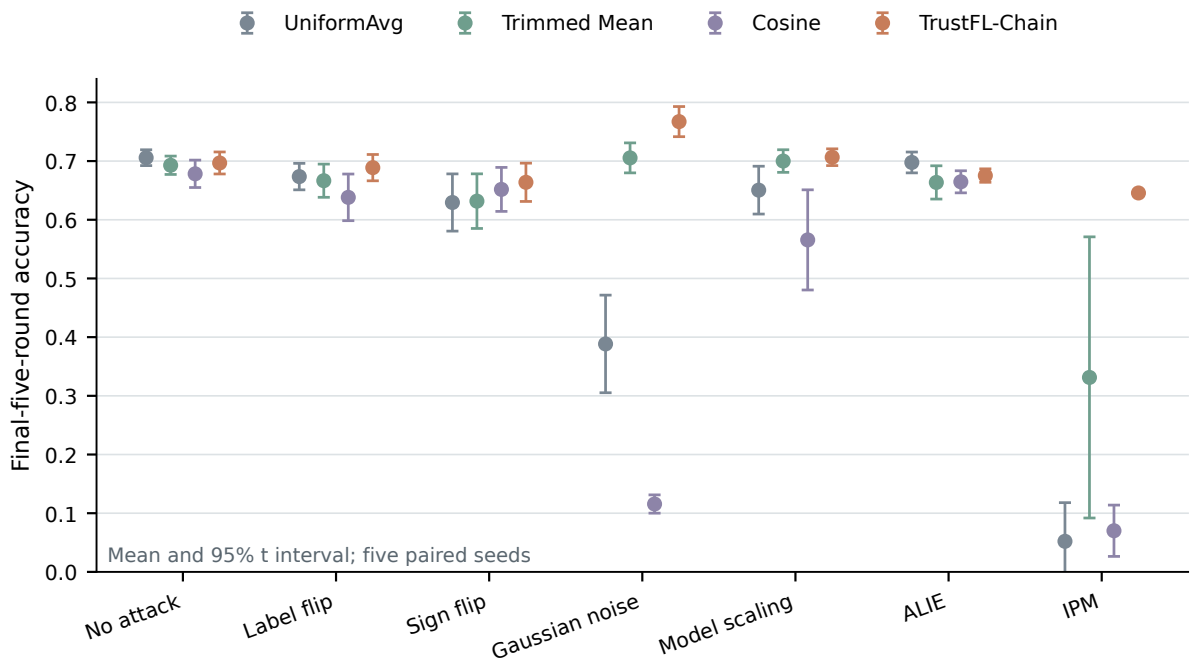

Figure 3. Fashion-MNIST final-five-round accuracy under Dirichlet non-IID $\alpha = 0.5$ and 20% malicious clients. Points show five-seed means and whiskers show 95% Student-t intervals. UniformAvg denotes equal client weights.

Table 4 gives the five-seed means alongside the strongest observed baseline for each attack and the corresponding TrustFL-Chain detection F1.

Label flipping produces a smaller positive mean difference and weaker detection F1. Here the attacker changes local training labels, which need not create a conspicuous submitted update. Sign flipping places TrustFL-Chain 0.0121 above Cosine on average, although the paired 95% interval $[-0.0006, 0.0248]$ crosses zero. Under ALIE, uniform averaging reaches 0.6977 and TrustFL-Chain reaches 0.6753. IPM shows the opposite pattern: TrustFL-Chain retains 0.6457 accuracy, compared with 0.3314 for the strongest observed baseline.

In the single-seed strength checks, ALIE final-five-round accuracy is 0.6997 at $z = 0.5$ and 0.6907 at $z = 1.0$; detection F1 remains low. IPM accuracy falls from 0.6594 at $\varepsilon = 1$ to 0.4726 at $\varepsilon = 10$, while detection F1 rises from 0.333 to 0.750 as the update becomes easier to spot.

Table 4. Main Fashion-MNIST results under Dirichlet non-IID $\alpha = 0.5$ and 20% malicious clients (five-seed means). Bold values indicate the higher accuracy in each row.

Attack	Best baseline	Baseline accuracy	TrustFL-Chain accuracy	TrustFL-Chain F1
No attack	Uniform averaging	**0.7058**	0.6968	–
Gaussian noise	Trimmed Mean	0.7054	**0.7672**	0.820
Model scaling	Trimmed Mean	0.7000	**0.7066**	0.842
Label flip	Uniform averaging	0.6736	**0.6888**	0.654
Sign flip	Cosine	0.6517	**0.6638**	0.471
ALIE	Uniform averaging	**0.6977**	0.6753	0.127
IPM	Trimmed Mean	0.3314	**0.6457**	0.704

5.2. RQ2: detection behavior

Figure 4 separates client-round from client-level detection. Uniform averaging and Trimmed Mean have no detection scores because they only aggregate updates. For TrustFL-Chain, client-round F1 reaches 0.842 under model scaling and 0.820 under Gaussian noise; it falls to 0.471 under sign flipping and 0.127 under ALIE.

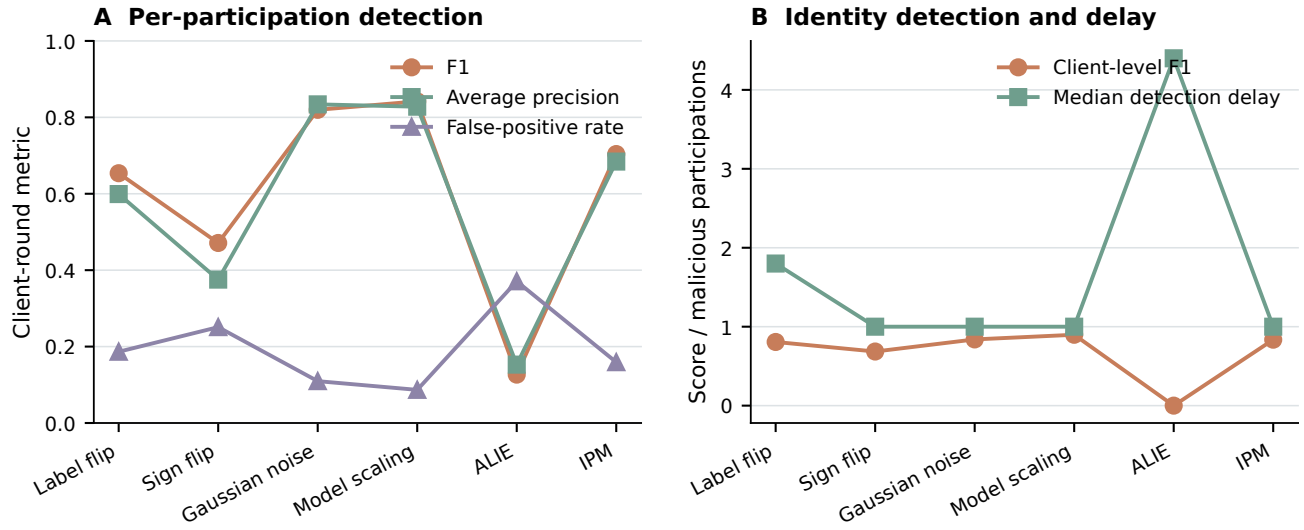

Figure 4. Detection behavior in the main Fashion-MNIST setting. Panel A reports client-round metrics. Panel B reports client-level F1 and median delay in malicious participations. Values are means over five seeds.

Table 5 reports pooled confusion counts and complementary metrics. True positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) are counted at the client-round level.

Figure 5 shows the six precision–recall curves summarized by average precision.

The paired Gaussian-noise difference from Trimmed Mean remains positive across its 95% interval (Table 6). Model scaling and sign flipping have positive mean differences, but their intervals cross zero.

Table 5. Detection results under Fashion-MNIST, Dirichlet $\alpha = 0.5$, and 20% malicious clients. TP, FP, FN, and TN pool five seeds at the client-round level; other entries are five-seed means.

Attack	TP	FP	FN	TN	F1	FPR	AP	Client F1	Delay
Label flip	256	223	49	972	0.654	0.186	0.599	0.806	1.8
Sign flip	187	300	118	895	0.471	0.251	0.375	0.685	1.0
Gaussian noise	303	131	2	1064	0.820	0.110	0.834	0.839	1.0
Model scaling	297	104	8	1091	0.842	0.087	0.828	0.898	1.0
ALIE	51	443	254	752	0.127	0.371	0.153	0.000	4.4
IPM	268	190	37	1005	0.704	0.159	0.684	0.836	1.0

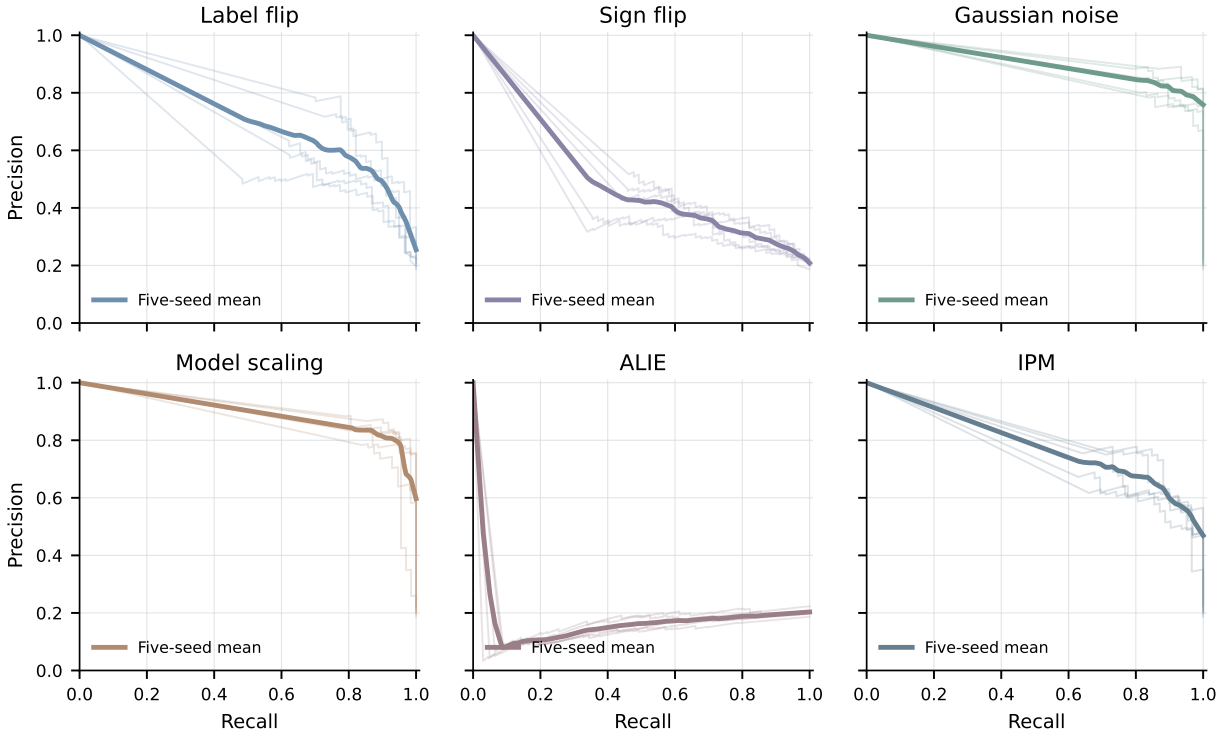

Figure 5. Client-round precision–recall curves for TrustFL-Chain under the six attacks. Thin lines show individual seeds; thick lines show recall-grid averages over seeds 42–46.

Table 6. Paired accuracy tests under Fashion-MNIST $\alpha = 0.5$. Positive differences favor TrustFL-Chain; g_z is paired Hedges' effect size. Confidence intervals are 95% Student-t intervals; p values use Holm adjustment.

Attack	Baseline	Difference	95% interval	g_z	p_{Holm}
Gaussian noise	Trimmed Mean	0.0618	[0.0524, 0.0712]	6.548	0.00021
Model scaling	Trimmed Mean	0.0065	[−0.0111, 0.0242]	0.368	0.362
Label flip	Uniform averaging	0.0152	[−0.0046, 0.0351]	0.762	0.201
Sign flip	Cosine	0.0121	[−0.0006, 0.0248]	0.944	0.173
ALIE	Trimmed Mean	0.0117	[−0.0078, 0.0312]	0.597	0.201
IPM	Cosine	0.5755	[0.5307, 0.6204]	12.757	< 0.0001

5.3. RQ3: ablations and malicious-ratio sensitivity

In Figure 6, removing reputation lowers model-scaling final accuracy from 0.710 to 0.592. A cosine-only detector falls further, to 0.469 accuracy and 0.086 mean F1. Removing the audit log leaves model metrics unchanged, as expected for a component that does not feed into training.

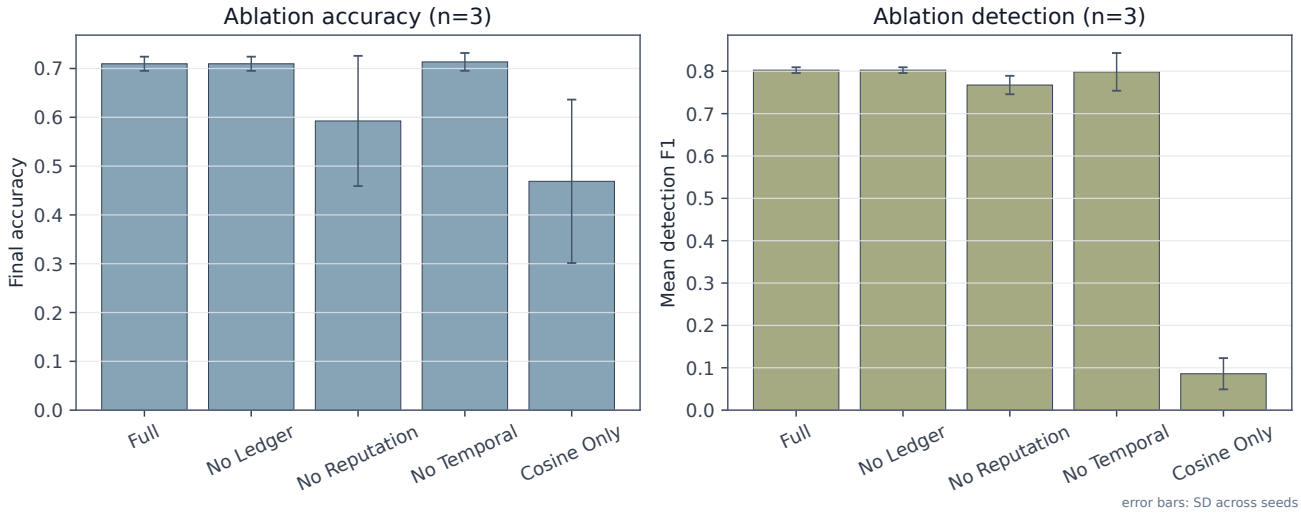

Figure 6. Ablation results under Fashion-MNIST, model scaling, non-IID $\alpha = 0.5$, and 20% malicious clients. Full TrustFL-Chain is shown as the reference. Reputation and multi-feature scoring contribute to robustness and detection. Removing the audit log does not change accuracy because it is not a training signal.

The lightweight History-Cosine comparator reaches 0.556 final accuracy and 0.241 mean F1 under model scaling. Under Gaussian noise, its final accuracy is near random. In this evaluated non-IID setting, its historical-similarity weighting falls short of the full method.

As the malicious ratio rises under model scaling, TrustFL-Chain remains competitive through 40% malicious clients (Figure 7). False positives also rise in the same sweep. At higher ratios, the method still needs enough honest updates to provide a usable learning signal.

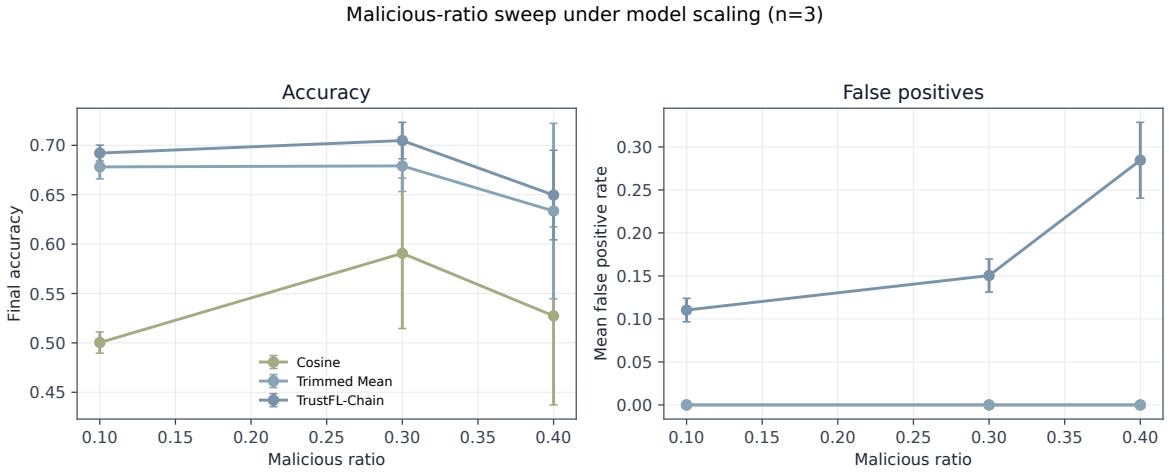

Figure 7. Malicious-ratio sweep under Fashion-MNIST model scaling with non-IID $\alpha = 0.5$. The left panel reports final accuracy; the right panel reports mean false positive rate. Error bars show SD across three seeds.

5.4. RQ3: parameter sensitivity and unequal client sizes

The one-factor model-scaling analysis appears in Figure 8. At the default settings, accuracy is 0.7066, F1 is 0.842, and FPR is 0.087. Lowering detector contamination from 0.2 to 0.1 raises F1 to 0.884 and lowers FPR to 0.054. Raising it to 0.3 yields F1 of 0.739 and FPR of 0.165. A penalty of 0.2 gives the highest observed accuracy, 0.7152; penalty 0.5 gives 0.7048. The main comparisons use the original default parameters.

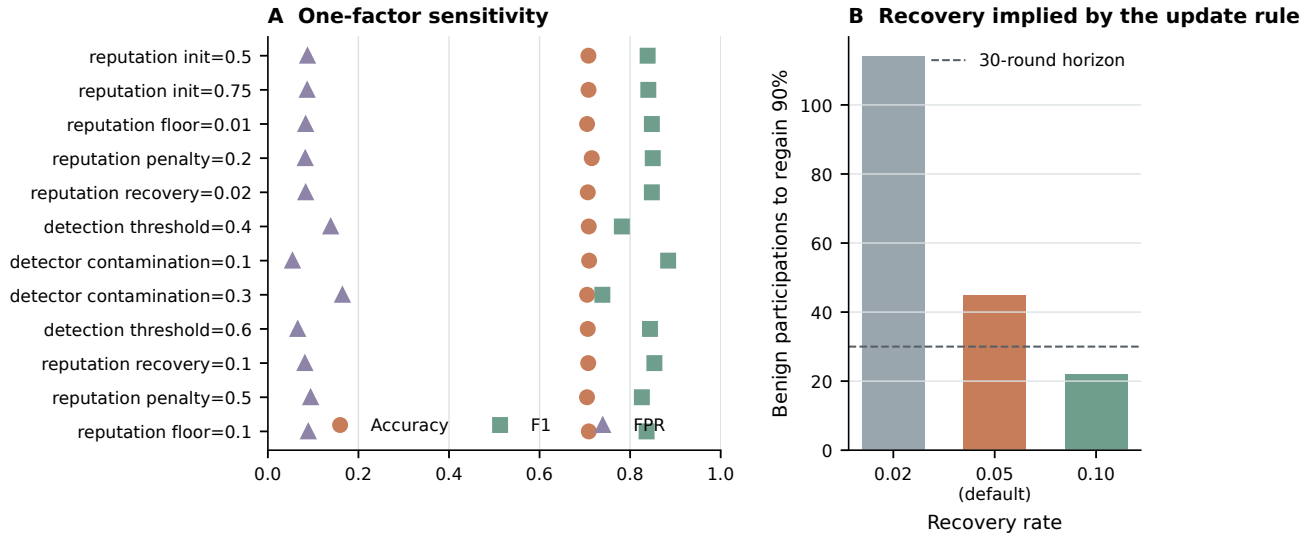

Figure 8. Parameter sensitivity and benign recovery under model scaling. Panel A changes one default parameter at a time and reports five-seed mean accuracy, F1, and FPR. Panel B gives benign participations required to regain 90% of one reputation penalty.

Within 30 rounds, no falsely penalized benign client completes 90% recovery; these observations are right-censored. At recovery rate 0.05, regaining 90% of one penalty requires 45 subsequent benign participations. Rates 0.02 and 0.10 require 114 and 22 participations, respectively. Recovery is built into the rule, but completion lies beyond the observed 30-round horizon.

Dirichlet partitioning produces unequal client sizes: a representative $\alpha = 0.5$ split ranges from 103 to 1002 examples. Without attack, uniform averaging and sample-weighted FedAvg reach 0.7058 and 0.7079. Under model scaling, they reach 0.6505 and 0.6547. Reputation-only TrustFL-Chain reaches 0.6968 without attack and 0.7066 under scaling; adding sample weighting gives 0.6984 and 0.7051. Sample counts therefore make little difference to the main scaling comparison in these partitions.

5.5. RQ4: supplemental transfer, overhead, and reputation

CIFAR-10 clean controls reach only 0.156–0.177 final-five-round accuracy under the 20-round small convolutional neural network (CNN) budget. Some attacked runs even exceed their clean counterparts: TrustFL-Chain reaches 0.2666 under Gaussian noise versus 0.1675 without attack. Undertraining and attack-induced optimization effects could explain this pattern. Figure 9 therefore serves as an optimization diagnostic, with no supported cross-dataset ranking.

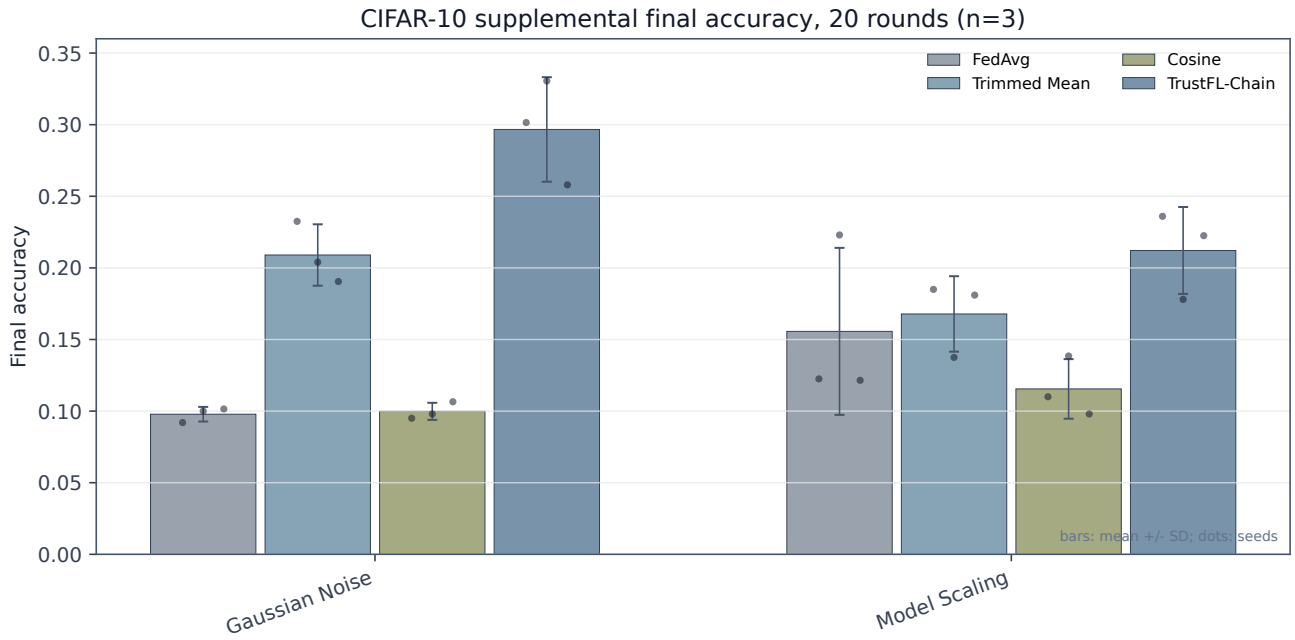

Figure 9. Exploratory CIFAR-10 attacked results with three seeds and 20 rounds under non-IID $\alpha = 0.5$. The low clean controls (0.156–0.177) indicate that this budget is undertrained; the figure is retained as a diagnostic rather than evidence of transfer.

End-to-end simulator measurements appear in Figure 10. A separate five-seed benchmark isolates defense overhead. Per round, uniform averaging takes 0.921 s, Cosine 0.930 s, Trimmed Mean 0.946 s, and TrustFL-Chain 1.068 s. In this single-machine benchmark, the proposed workflow is 15.9% slower than the uniform control. After 30 rounds, the audit file occupies roughly 54–56 KiB, or about 1.8 KiB per round. The timing excludes network latency and consensus costs.

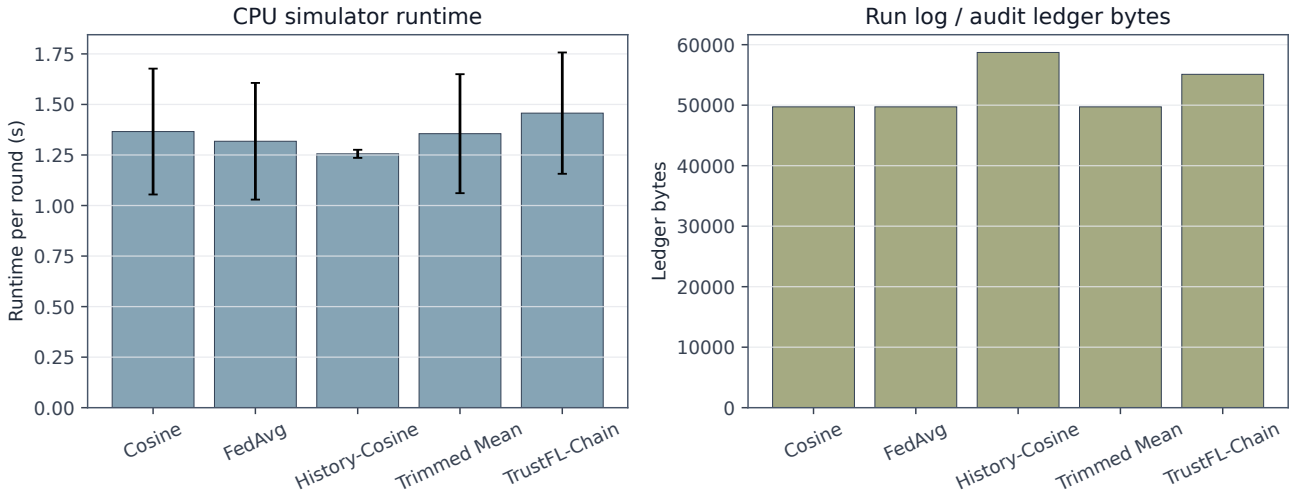

Figure 10. Runtime and audit-log overhead in the main Fashion-MNIST setting. Runtime is measured in the single-machine simulator. Logged bytes include common run records and the audit metadata used by TrustFL-Chain.

In the representative trajectory (Figure 11), malicious clients tend to lose reputation while benign clients stay nearer the upper range. The groups still overlap, which motivates soft weighting over permanent exclusion.

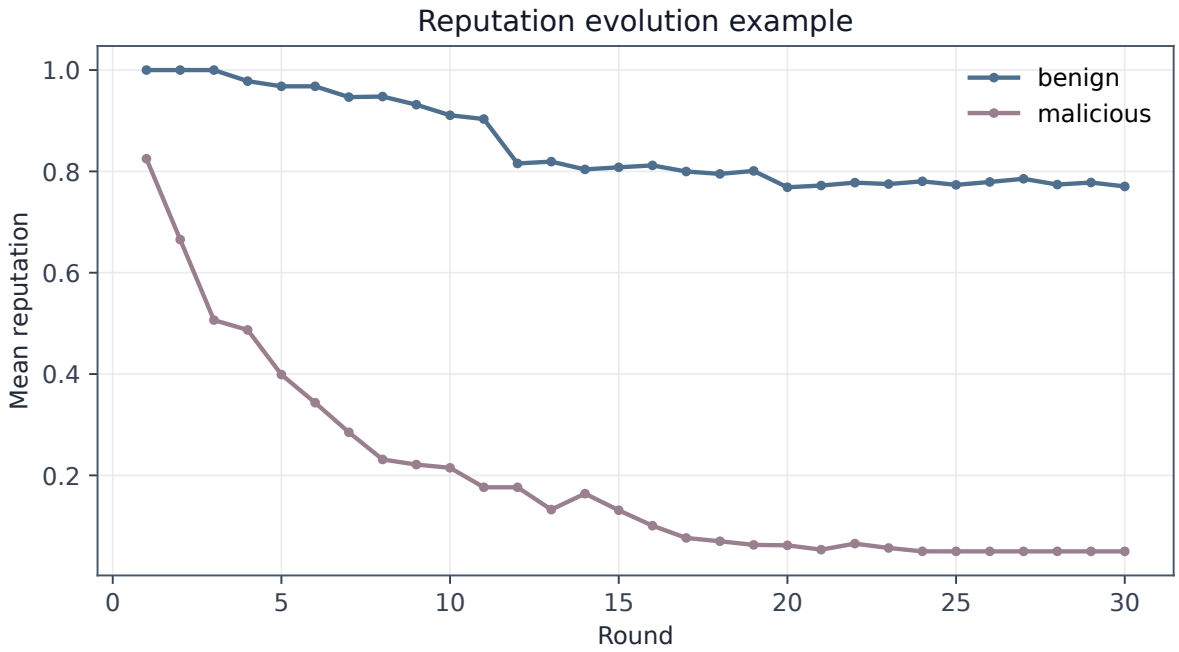

Figure 11. Representative reputation trajectory under model scaling. Malicious clients tend to lose reputation over time, while benign clients usually remain higher. The separation is imperfect, motivating soft weighting rather than permanent exclusion.

6. Discussion, limitations, and conclusion

Results differ by attack. TrustFL-Chain achieves its strongest combination of accuracy and detection under Gaussian noise and IPM. Under sign flipping, its mean accuracy exceeds Cosine by 1.21 percentage points, but the paired confidence interval crosses zero. ALIE is harder to identify: detection F1 is low, and final accuracy falls below uniform averaging. Together, these results show that reputation weighting depends on risk scores that separate malicious updates from ordinary variation.

To examine sign flipping further, we tested a stricter direction filter. In a focused rerun, its accuracy reached 0.462, above 0.434 for Cosine in the same rerun. Its Gaussian-noise and model-scaling accuracies, however, fell to 0.089 and 0.402. Table 7 lists the original artifact result separately from two reruns. The filter improves one focused comparison without producing a more balanced result, so the main experiments retain the original aggregation rule.

Table 7. Direction-filter check under Fashion-MNIST, non-IID $\alpha = 0.5$, and 20% malicious clients. The original artifact result and the two separate reruns are shown in distinct rows, not as a paired comparison. The filter improves the focused sign-flip comparison against a current Cosine rerun but weakens Gaussian-noise and model-scaling robustness. Bold values mark the original artifact result, not the best value in each column.

Setting	Noise	Label	Scaling	Sign
Original artifact result	**0.782**	**0.712**	**0.710**	**0.654**
Reputation-weighted rerun	0.698	0.392	0.542	0.345
Direction-filter rerun	0.089	0.437	0.402	0.462

The audit log preserves update hashes, risk scores, reputation, and aggregation decisions for each round. Comparison with a previously retained hash or external copy can expose later edits; a file held only by the server cannot prevent that server from rewriting the entire record. Its auditability therefore depends on how the hash is retained.

The method also requires the server to read each client's update. Combining risk scoring with secure aggregation, which hides individual updates, would require a different design. Trusted execution, secure multiparty computation, homomorphic computation of limited statistics, or privacy-preserving proofs are possible directions, each with different computational costs and trust assumptions.

The experiments use a single-machine, synchronous simulator rather than a deployed network or asynchronous training system. They do not examine attackers who adapt to the detector's internal features or quantify privacy leakage from inspecting updates. CIFAR-10 remains undertrained within the available budget and does not support a cross-dataset robustness ranking. Five seeds limit statistical precision; the default detector uses a prior matching the main malicious ratio, with only moderate mismatch tested. History-Cosine is FoolsGold-like rather than a strict reproduction, and the local log has no independent anchor.

Across 440 validated jobs, TrustFL-Chain shows its clearest gains under Gaussian noise and IPM. Ablations support the roles of reputation and multi-feature scoring, while the malicious-ratio sweep shows rising false positives. Label flipping, sign flipping, and ALIE remain harder cases. For the tested direct update manipulations, the recovery rule allows client weights to rise after subsequent low-risk updates, while the local audit record makes aggregation decisions easier to check after training.

Data availability statement

The implementation and experiment scripts are available at https://github.com/TeacherMK/trustfl-chain. The public repository contains source code and tests, but not generated experiment outputs or figures. Fashion-MNIST and CIFAR-10 are publicly available datasets. The source files for this manuscript version accompany the resubmission.

Declaration of generative AI and AI-assisted technologies

The authors conceived the study, prepared the initial manuscript draft, and conducted and interpreted the experiments. OpenAI Codex and ChatGPT assisted with translation, English-language editing throughout the manuscript, and paragraph-level rewriting of author-provided text. Codex also provided coding support and assisted with repository preparation. The authors subsequently revised the manuscript by hand, reviewed the final text, and take full responsibility for its content.

Acknowledgements

The authors thank the reviewers for their comments on an earlier version of this manuscript.

Authors' contribution

Conceptualization, Z.Z. and D.L.; methodology, Z.Z. and H.L.; software, Z.Z.; validation, Z.Z., P.M., H.L. and L.F.; formal analysis, Z.Z., P.M. and H.L.; investigation, Z.Z. and P.M.; data curation, Z.Z.; writing—original draft preparation, Z.Z.; writing—review and editing, Z.Z., D.L., P.M., H.L. and L.F.; visualization, Z.Z.; supervision, D.L.; project administration, Z.Z. All authors have read and agreed to the published version of the manuscript.

Conflicts of interest

The authors declare no competing interests.

References

[1] McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, USA, April 20–22, 2017, pp. 1273–1282.

[2] Kairouz P, McMahan HB. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* 2021, 14(1–2):1–210.

[3] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: concept and applications. *ACM Trans. Intell. Syst. Technol.* 2019, 10(2):1–19.

[4] Li T, Sahu AK, Talwalkar A, Smith V. Federated learning: challenges, methods, and future directions. *IEEE Signal Process Mag.* 2020, 37(3):50–60.

[5] Bhagoji AN, Chakraborty S, Mittal P, Calo S. Analyzing federated learning through an adversarial lens. In *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, USA, June 9–15, 2019, pp. 634–643.

[6] Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V. How to backdoor federated learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, Online, August 26–28, 2020, pp. 2938–2948.

[7] Sun Z, Kairouz P, Suresh AT, McMahan HB. Can you really backdoor federated learning? *arXiv* 2019, arXiv:1911.07963.

[8] Baruch G, Baruch M, Goldberg Y. A little is enough: circumventing defenses for distributed learning. In *Proceedings of Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, Vancouver, Canada, December 8–14, 2019.

[9] Konečný J, McMahan HB, Ramage D, Richtárik P. Federated optimization: distributed machine learning for on-device intelligence. *arXiv* 2016, arXiv:1610.02527.

[10] Zhao Y, Li M, Lai L, Suda N, Civin D, *et al.* Federated learning with non-IID data. *arXiv* 2018, arXiv:1806.00582.

[11] Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, *et al.* Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems 2 (MLSys 2020)*, Austin, USA, March 2–4, 2020, pp. 429–450.

[12] Karimireddy SP, Kale S, Mohri M, Reddi SJ, Stich SU, *et al.* SCAFFOLD: stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*, Online, July 13–18, 2020, pp. 5132–5143.

[13] Reddi SJ, Charles Z, Zaheer M, Garrett Z, Rush K, *et al.* Adaptive federated optimization. *arXiv* 2021, arXiv:2003.00295.

[14] Wang J, Liu Q, Liang H, Joshi G, Poor HV. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Proceedings of Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, Online, December 6–12, 2020, pp. 7611–7623.

[15] Haddadpour F, Mahdavi M. On the convergence of local descent methods in federated learning. *arXiv* 2019, arXiv:1910.14425.

[16] Blanchard P, El Mhamdi EM, Guerraoui R, Stainer J. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Long Beach, USA, December 4–9, 2017.

[17] Yin D, Chen Y, Kannan R, Bartlett P. Byzantine-robust distributed learning: towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, July 10–12, 2018, pp. 5650–5659.

[18] Chen Y, Su L, Xu J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, Irvine, USA, June 18–22, 2018, pp. 1–25.

[19] El Mhamdi EM, Guerraoui R, Rouault S. The hidden vulnerability of distributed learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, July 10–15, 2018, pp. 3521–3530.

[20] Xie C, Koyejo S, Gupta I. Generalized Byzantine-tolerant SGD. *arXiv* 2018, arXiv:1802.10116.

[21] Pillutla K, Kakade SM, Harchaoui Z. Robust aggregation for federated learning. *IEEE Trans. Signal Process.* 2022, 70:1142–1154.

[22] Fung C, Yoon CJM, Beschastnikh I. The limitations of federated learning in sybil settings. In *Proceedings of 23rd International Symposium on Research in Attacks, Intrusions and Defenses*, San Sebastian, Spain, October 14–16, 2020, pp. 301–316.

[23] Cao X, Fang M, Liu J, Gong NZ. FLTrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv* 2020, arXiv:2012.13995.

[24] Kim H, Park J, Bennis M, Kim SL. Blockchained on-device federated learning. *arXiv* 2020, arXiv:1808.03949.

[25] Qu Y, Uddin MP, Gan C, Xiang Y, Gao L, *et al.* Blockchain enabled federated learning: a survey. *ACM Comput. Surv.* 2022, 55(4):1–35.

[26] Nguyen DC, Ding M, Pham QV, Pathirana PN, Le LB, *et al.* Federated learning meets blockchain in edge computing: opportunities and challenges. *IEEE Internet Things J.* 2021, 8(16):12806–12825.

[27] Lu Y, Huang X, Dai Y, Maharjan S, Zhang Y. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. *IEEE Trans. Ind. Inf.* 2020, 16(6):4177–4186.

[28] Shayan M, Fung C, Yoon CJM, Beschastnikh I. Biscotti: a blockchain system for private and secure federated learning. *IEEE Trans. Parallel Distrib. Syst.* 2021, 32(7):1513–1525.

[29] Li D, Han D, Crespi N, Minerva R, Raza SM, *et al.* Blockchain in the digital twin context: a comprehensive survey. *ACM Comput. Surv.* 2025, 58(6):1–34.

[30] Liu Y, Xing X, Cheng H, Li D, Guan Z, *et al.* A flexible sharding blockchain protocol based on cross-shard Byzantine fault tolerance. *IEEE Trans. Inf. Forensics Secur.* 2023, 18:2276–2291.

[31] Liu Y, Xing X, Tong Z, Lin X, Chen J, *et al.* Secure and scalable cross-domain data sharing in zero-trust cloud-edge-end environment based on sharding blockchain. *IEEE Trans. Dependable Secure Comput.* 2024, 21(4):2603–2618.

[32] Li D, Li H, Crespi N, Minerva R, Li M, *et al.* Hyper-IIoT: a smart contract-inspired access control scheme for resource-constrained industrial internet of things. *IEEE Trans. Sustainable Comput.* 2025, 10(5):820–829.

[33] Li D, Crespi N, Minerva R, Liang W, Li KC, *et al.* DPS-IIoT: non-interactive zero-knowledge proof-inspired access control towards information-centric industrial internet of things. *Comput. Commun.* 2025, 233:108065.

[34] Chai Z, Ali A, Zawad S, Truex S, Anwar A, *et al.* TiFL: a tier-based federated learning system. In *Proceedings of The 29th International Symposium on High-Performance Parallel and Distributed Computing*, Stockholm, Sweden, June 23–26, 2020, pp. 125–136.

[35] Reisizadeh A, Mokhtari A, Hassani H, Jadbabaie A, Pedarsani R. FedPAQ: a communication-efficient federated learning method with periodic averaging and quantization. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, Online, August 26–28, 2020, pp. 2021–2031.

[36] Fang M, Cao X, Jia J, Gong NZ. Local model poisoning attacks to Byzantine-robust federated learning. In *Proceedings of 29th USENIX Security Symposium*, Boston, USA, August 12–14, 2020, pp. 1605–1622.

[37] Xie C, Koyejo S, Gupta I. Fall of empires: breaking Byzantine-tolerant SGD by inner product manipulation. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, Online, August 3–6, 2020, pp. 261–270.

[38] Tolpegin V, Truex S, Gursoy ME, Liu L. Data poisoning attacks against federated learning systems. In *Proceedings of European Symposium on Research in Computer Security*, Online, September 14–18, 2020, pp. 480–501.

[39] Wang H, Sreenivasan K, Rajput S, Vishwakarma H, Agarwal S, *et al.* Attack of the tails: yes, you really can backdoor federated learning. In *Proceedings of Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, Online, December 6–12, 2020, pp. 16070–16084.

[40] Özdayi MS, Kantarcioglu M, Gel EA. Defending against backdoors in federated learning with robust learning rate. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Online, February 2–9, 2021, pp. 9268–9276.

[41] Andreina S, Marson GA, Möller H, Karame G. BAFFLE: backdoor detection via feedback-based federated learning. In *Proceedings of 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, Washington, USA, July 7–10, 2021, pp. 852–863.

[42] Rieger P, Nguyen TD, Miettinen M, Sadeghi AR. DeepSight: mitigating backdoor attacks in federated learning through deep model inspection. *arXiv* 2022, arXiv:2201.00763.

[43] Shejwalkar V, Houmansadr A. Manipulating the Byzantine: optimizing model poisoning attacks and defenses for federated learning. In *Proceedings of 28th Annual Network and Distributed System Security Symposium (NDSS 2021)*, Online, February 21–25, 2021.

[44] Jia J, Liu Y, Cao X, Gong NZ. Certified robustness of nearest neighbors against data poisoning and backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vancouver, Canada, February 22–March 1, 2022, pp. 9575–9583.

[45] Nguyen TD, Rieger P, Chen H, Yalame HF, Möllering H, *et al.* FLAME: taming backdoors in federated learning. *arXiv* 2021, arXiv:2101.02281.

[46] Mothukuri V, Parizi RM, Pouriyeh S, Huang Y, Dehghantanha A, *et al.* A survey on security and privacy of federated learning. *Future Gener. Comput. Syst.* 2021, 115:619–640.

[47] Lyu L, Yu H, Yang Q. Threats to federated learning: a survey. *arXiv* 2020, arXiv:2003.02133.

[48] Zhu L, Liu Z, Han S. Deep leakage from gradients. In *Proceedings of Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, Vancouver, Canada, December 8–14, 2019.

[49] 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 *Proceedings of 2019 IEEE Symposium on Security and Privacy (SP)*, San Francisco, USA, May 19–23, 2019, pp. 739–753.

[50] Geyer RC, Klein T, Nabi M. Differentially private federated learning: a client level perspective. *arXiv* 2017, arXiv:1712.07557.

[51] Basak S, Kumar A, Chatterjee K. Blockchain enabled federated learning-based medical cyber-physical system. In *Proceedings of 2024 First International Conference on Electronics, Communication and Signal Processing (ICECSP)*, New Delhi, India, August 8–10, 2024, pp. 1–6.

[52] Kumar A, Chatterjee K. A lightweight blockchain-based framework for medical cyber-physical system. *J. Supercomput.* 2023, 79(11):12013–12041.

[53] Kumar A, Chatterjee K. Securing internet of medical devices using energy efficient blockchain for healthcare 4.0. *Cluster Comput.* 2024, 27(6):8333–8348.

[54] Kumar A, Chatterjee K, Singh A, Kumar A, Aldawsari H. Privacy-preserving federated learning with blockchain auditing for consumer electronics applications. *IEEE Trans. Consum. Electron.* 2026, 72(1):1554–1561.

[55] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, *et al.* Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas, USA, October 30–November 3, 2017, pp. 1175–1191.