Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Secure-Aggregation Federated Learning with Tamper-Evident Logging for Edge Anomaly Detection in Smart Healthcare

Gangarapu Nagalakshmi

nagalakshmics804@gmail.com

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology

John T. Mesia Dhas

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology

Charlesprabhu V

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology Chennai, India - 600062

Research Article

Keywords: Federated learning, secure aggregation, tamper-evident logging, Merkle tree, anomaly detection, Internet of Medical Things, edge computing, smart healthcare

Posted Date: November 24th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11490553/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Secure-Aggregation Federated Learning with Tamper-Evident Logging for Edge Anomaly Detection in Smart Healthcare

Gangarapu Nagalakshmi, John T. Mesia Dhas and Charles Prabhu V

[1]Research Scholar,Department of Computer Science and Engineering, School of Computing, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India.

[2]Associate Professor,Department of Artificial Intelligence & Data Science, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India.

[3]Assistant Professor, Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India.

Corresponding author email: nagalakshmics804@gmail.com

***Abstract*—**Internet of Medical Things (IoMT) networks must detect anomalies close to the patient, yet privacy rules prevent pooling their data centrally. Federated learning (FL) keeps data on edge devices, but model updates can still leak information and the aggregation process cannot be audited afterwards. This paper proposes SATEL-FL, a federated learning framework that combines dropout-tolerant secure aggregation with a tamper-evident training log for edge anomaly detection in smart healthcare. Clients mask their updates with pairwise elliptic-curve Diffie–Hellman keys, so the server learns only the weighted sum, and Shamir secret sharing removes the masks of clients that drop out. Each round, hashes of the masked updates, the aggregate and the new model are appended to a Merkle-tree log whose signed root is checked by an independent witness. Evaluated in MATLAB on WUSTL-EHMS-2020 with identifier columns removed to prevent label leakage, SATEL-FL reaches an F1-score of 67.5 ± 0.9% and a ROC-AUC of 0.882 ± 0.005 over five seeds, matching plain FedAvg, within 3.8 F1 points of centralised training, and robust to 30% client dropout. With 10 clients, secure aggregation costs 22–48 ms per client per round and logging 8–16 ms. All 800 injected log-tampering attacks were detected, and compared with PBFT blockchain logging the log cuts latency by 4.8–6.6× and storage by 5.4–17.5×. On CICIoMT2024 the federated detector reached an F1-score of 89.4% and a ROC-AUC of 0.985. Secure aggregation alone does not resist model poisoning, which motivates robust secure aggregation.

Index Terms*—Federated learning, secure aggregation, tamper-evident logging, Merkle tree, anomaly detection, Internet of Medical Things, edge computing, smart healthcare.*

I. INTRODUCTION

Smart healthcare increasingly relies on the Internet of Medical Things (IoMT): wearable sensors, bedside monitors, implantable devices and gateways that continuously acquire physiological signals and transmit them for clinical analysis [1], [2]. Anomalies in these streams may indicate a deteriorating patient, a faulty sensor or a cyber-attack on the medical network, and they must be detected with low latency. This requirement pushes analytics from the cloud towards edge and fog nodes located close to the patient [3]–[6]. At the same time, medical records are among the most sensitive categories of personal data, and regulations such as HIPAA, the EU General Data Protection Regulation and India's Digital Personal Data Protection Act, 2023 restrict how such data may be collected, shared and processed.

Federated learning (FL) addresses this tension by training a shared model across many clients while the raw data remains on the device that produced it; only model updates are sent to an aggregation server [1], [7], [8]. FL has been applied to anomaly detection in digital health [9], [10], in IoT–edge environments [11]–[13], to multivariate time series [14] and to other cyber-physical domains such as

smart grids and additive manufacturing [15], [16]. In these settings it generally approaches the accuracy of centralised training with much lower privacy exposure, and communication-efficient variants further reduce the cost of exchanging updates [17].

However, FL is not secure by default. Shared gradients and model updates can be inverted to reconstruct training samples or to infer whether a particular patient's record was used, and compromised clients can submit poisoned updates that degrade the global model or implant backdoors [18]–[20]. Secure aggregation (SecAgg) protocols address the first problem by ensuring that the server learns only the aggregate of the client updates, not any individual contribution [21]–[24]. SecAgg alone, however, does not meet a second requirement of clinical systems: accountability. After a training round, hospitals, regulators and auditors have no reliable way to verify which clients participated, which updates were committed, or whether the sequence of global models used for clinical decisions has been altered afterwards. Such changes could come from an external attacker or from an insider with access to the aggregation server.

Most existing work provides such integrity guarantees with blockchain. Blockchain-based FL frameworks record model updates or aggregation results on a distributed ledger for healthcare [25]–[28], for IoMT intrusion and malicious-node detection [29], [30] and for edge anomaly detection [31], [32], sometimes combined with trusted execution environments [33] or lightweight consensus [34]. Other work relies on cryptographic verification such as zero-knowledge proofs and fully homomorphic encryption [35], [36]. These approaches provide strong guarantees, but their consensus latency, ledger replication and proof-generation costs are difficult to reconcile with battery-powered, memory-constrained medical edge devices [2], [8]. Many of them also treat privacy-preserving aggregation and auditability as separate layers: the ledger either stores unmasked updates, which partly defeats secure aggregation, or stores only the final model and cannot show how it was produced.

Tamper-evident logging is a lighter alternative. Hash-chained and Merkle-tree logs, originally developed for secure audit trails [37], [38], let any verifier who holds a small signed commitment detect modification, deletion or reordering of past records with proofs of logarithmic size, without consensus among many nodes. Combining such a log with secure aggregation suits healthcare FL well: the log binds each round to the hashes of the masked updates and to the resulting aggregate, giving an auditable training history, while the masks continue to hide individual updates. Recent audited and verifiable FL schemes for edge anomaly detection and e-health [20], [39], [40] point in this direction. To the best of our knowledge, however, no existing framework jointly provides dropout-tolerant secure aggregation, lightweight tamper-evident logging and edge-deployable anomaly detection for smart healthcare. A recent systematisation of verifiable FL [41] likewise reports no scheme that records masked updates in a Merkle-tree log.

This paper proposes SATEL-FL (Secure-Aggregation, Tamper-Evident-Logging Federated Learning) to fill this gap. The main contributions are as follows:

1) A system and threat model for federated anomaly detection in smart healthcare covering an honest-but-curious aggregator, network adversaries, malicious clients and an insider who attempts to alter, delete, reorder or roll back training records.
2) A dropout-tolerant secure-aggregation protocol that applies pairwise additive masking [21] to weighted Federated Averaging [7], using P-256 elliptic-curve Diffie–Hellman key agreement, an AES-128-CTR mask generator and t-of-K Shamir secret sharing, with fixed-point arithmetic that is exact in MATLAB.
3) A tamper-evident training log that commits every masked upload, aggregate and global model to an append-only Merkle tree with hash-chained, ECDSA-signed tree heads verified by an

independent witness, giving clients logarithmic-size inclusion proofs and auditors logarithmic-size consistency proofs.

4) A lightweight neural detector (a 35-32-16-1 multilayer perceptron with 1,697 parameters) evaluated on WUSTL-EHMS-2020 under a leakage-free protocol, together with evidence, consistent with a concurrent leakage audit [42], that identifier columns in this dataset make near-perfect accuracy trivially achievable. A similar check finds three such artifact features in CICIoMT2024 [43].
5) A MATLAB evaluation over five seeds covering detection performance, non-IID data, 10–30% client dropout, 10–100 clients, eight log-tampering attacks plus a no-witness ablation, a PBFT blockchain-logging baseline and two poisoning attacks, complemented by a validation of detection, non-IID data, dropout and poisoning on CICIoMT2024.

The rest of this paper is organised as follows. Section II reviews related work. Section III introduces the building blocks, and Section IV defines the system and threat models. Section V presents SATEL-FL, and Section VI analyses its security. Section VII describes the experimental setup, Section VIII reports and discusses the results, and Section IX concludes.

II. RELATED WORK

This section reviews prior work in five areas: federated learning frameworks for smart healthcare, federated anomaly and intrusion detection, secure and verifiable aggregation, blockchain-based integrity for healthcare FL, and tamper-evident logging. It ends with a comparison of representative schemes and a statement of the research gap.

A. Federated Learning Frameworks for Smart Healthcare

Rahman et al. [1] survey FL-based AI approaches in smart healthcare and set out a taxonomy of architectures, applications and open challenges, highlighting privacy leakage, data heterogeneity and system security as recurring problems. Alhawas and Rassam [18] survey vulnerabilities, attacks and defences for FL in edge computing, covering inference attacks, poisoning and Byzantine behaviour, and argue that defences must be matched to the resource limits of edge nodes. Mahmood et al. [2] propose an adaptive, resource-aware federated edge learning framework for real-time IoMT applications. It combines adaptive modular learning units, dynamic data encoding, hierarchical device–regional–global aggregation and privacy-preserving secure enclaves with differential-privacy noise, and reports 96.3% accuracy with 110 ms streaming anomaly-detection latency. Their framework protects updates with enclaves and noise but does not maintain a verifiable record of the aggregation process.

Several frameworks target the edge tier specifically. Shokri [4] studies privacy-preserving edge FL for intelligent mobile-health systems, and Akter et al. [8] propose SPEI-FL, a serverless privacy edge-intelligence FL scheme that removes the dependence on a single central aggregator in smart healthcare. Rajagopal et al. [3] present FedSDM, an FL-based decision-making module for ECG data across edge, fog and cloud layers. Maurya et al. [44] use secure edge aggregation for privacy-preserving severity classification. Liu et al. [45] design a secure and efficient FL-based smart healthcare system, and Khan et al. [46] perform encrypted aggregation for wearable health devices inside programmable network elements. Together, these works show that edge-centric FL is feasible in healthcare, but they focus on privacy and efficiency rather than on making the training history auditable.

B. Federated Anomaly and Intrusion Detection

Anomaly detection is a natural fit for FL because anomalies are rare and scattered across sites. Raza et al. [9] propose AnoFed, which combines transformer-based federated learning with support vector data description for adaptive anomaly detection in digital health. Gupta et al. [10] use hierarchical FL with digital twins for anomaly detection in smart healthcare. Xiang et al. [11] adapt isolation forest to federated training in the IoT–edge continuum, and Raj et al. [12] address privacy-preserving anomaly detection in heterogeneous IoT networks. Al-Alshaikh [13] proposes an energy-aware FL anomaly detection algorithm for big-data analytics in IoT-based smart healthcare, which shows that energy is a primary constraint at the medical edge. Zhang et al. [14] present FAPAD, which optimises federated aggregation for privacy-preserving time-series anomaly detection, and Xu et al. [47] apply privacy-preserving federated deep learning to robust anomaly detection in distributed security-sensing systems. Outside healthcare, Jithish et al. [15] demonstrate FL-based anomaly detection in smart grids, and Mahdi et al. [16] perform secure multi-modal federated anomaly detection for wire arc additive manufacturing. Khan et al. [17] propose a collaborative SRU network with dynamic behaviour aggregation and explainable features that reduces communication overhead.

Edge-AI architectures for real-time patient monitoring have also been proposed. Karpagam et al. [5] and Prabha et al. [6] integrate edge-AI with secure IoT and wireless architectures for real-time healthcare monitoring and federated anomaly detection. On the network-security side, Bensaid et al. [48] propose SA-FLIDS, a secure and authenticated FL-based intrusion detection system for smart healthcare. Sharma et al. [49] propose a privacy-enhanced FL framework for intrusion detection in smart IoT environments, and Sengan and Shieh [50] combine non-IID-aware federated intrusion detection with PBFT consensus for secured aggregation across multi-institutional healthcare IoT networks. Tang [51] presents RedPO-BRNNet, which combines differential privacy and secure aggregation for federated anomaly detection at the IoT edge. Azzan and Alkholidi [39] propose SEAF-FL-MLP, an audited federated edge-AI anomaly detector that also analyses client heterogeneity and shortcut sensitivity. It is one of the few works that treat auditing as a first-class requirement.

C. Secure and Verifiable Aggregation

Secure aggregation was made practical for FL by Bonawitz et al. [21], whose protocol uses pairwise masks that cancel in the sum, together with Shamir secret sharing to tolerate client dropout. Later work extends this protocol in several directions. Zhang et al. [22] propose SAFELearning, which keeps aggregation secure while making backdoor attacks detectable. Yao et al. [23] and Zhang et al. [24] (VSAF) design verifiable secure aggregation schemes that let clients check that the server returned a correct aggregate, the latter specifically for edge computing. Al-Mashakbah et al. [52] introduce a reputation-aware secure aggregation protocol for IoMT intrusion detection. Liu and Wu [53] propose quantum-resistant secure aggregation for healthcare FL, and Rajabi and Toorani [54] apply secure aggregation to clinical EEG data.

Heavier cryptographic tools have also been used. Othman and Mihret [35] combine fully homomorphic encryption with zk-SNARK-enabled edge aggregation for scalable IoMT analytics, and Reis [36] uses zero-knowledge proofs to make privacy verifiable in federated anomaly detection for 5G edge-IoT systems. Basudan and Alamer [40] remove the central server entirely with a verifiable, privacy-preserving peer-to-peer FL scheme with collaborative incentives for edge-IoMT. To counter poisoning, Kalapaaking et al. [55] verify models with secure multi-party computation, Liu et al. [19] propose PnA, a robust aggregation rule against poisoning attacks for edge intelligence, and Alruwaili and Moulahi [20] present a robust and verifiable FL framework that prevents data-poisoning threats in e-health. VerifyNet [56] and VeriFL [57] let clients check the aggregate returned by the server with homomorphic

hashes and commitments. These schemes verify the correctness of a single aggregation but generally do not keep a persistent, append-only record that can be audited across rounds.

D. Blockchain-Based Integrity for Healthcare FL

Blockchain is the most common way to add integrity and traceability to FL. Chang et al. [25] propose an early blockchain-based FL method for smart healthcare. Kalapaaking et al. combine blockchain with secure aggregation in a trusted execution environment for IoT [33] and propose smart policy control for securing the FL management system [58]. M. S. and J. R. [26] integrate blockchain-enabled FL with edge analytics for electronic health record management, and Nawaz et al. [27] use blockchain with privacy-preserving second-order federated edge learning for personalised healthcare. Split learning has also been combined with blockchain: Baihan et al. [28] propose a cloud-assisted, blockchain-enabled split FL framework for IoMT in Healthcare 5.0, and Babar et al. [29] present TAWB-SFL, a trust-aware blockchain-orchestrated split FL scheme for intrusion detection in 6G healthcare IoT.

Other works apply blockchain to detection tasks directly. Lilhore et al. [31] pair blockchain-enabled FL with a hybrid CNN-LSTM anomaly detector for edge IoT networks. Huang and Ren [32] integrate blockchain and FL for distributed anomaly detection and protection, and Sonani et al. [30] use blockchain-enabled FL to detect malicious nodes in decentralised healthcare networks. Rajagopal and Natesan [59] propose PEF-CAPD, a privacy-enhanced federated cyber-physical attack detection framework for edge–cloud–blockchain smart healthcare. Recognising the overhead of conventional ledgers, Melhem et al. [34] adopt a lightweight blockchain with clustered FL and adaptive model selection. Across this body of work, the integrity layer is typically a full or permissioned ledger whose consensus, storage and communication costs grow with the number of participants and rounds, which is a significant burden for constrained medical edge devices [2].

E. Tamper-Evident Logging

Tamper-evident logging predates FL. Schneier and Kelsey [37] introduced secure audit logs in which each entry is cryptographically chained to its predecessor, so that an attacker who compromises the logging machine cannot undetectably alter entries written before the compromise. Crosby and Wallach [38] replaced the linear hash chain with a history tree, a Merkle-tree-based structure that supports logarithmic-size membership and consistency proofs. This allows auditors to check that a log has only ever been appended to. The same idea underlies Certificate Transparency on the web. Such logs need only a trusted or replicated signed root rather than global consensus, which makes them attractive for edge systems. BlockFLA [60] makes FL accountable by committing hashes of plaintext updates to a public blockchain, so that cheating clients can be identified afterwards. A recent systematisation of verifiable FL [41] finds no scheme that records masked updates in a Merkle-tree transparency log. The use of such logs for auditing secure-aggregation FL in healthcare has therefore received little attention.

F. Comparative Analysis and Research Gap

Table I compares representative schemes along the dimensions most relevant to this work.

TABLE I. Comparison of Representative Secure and Verifiable FL Schemes

Ref.	Domain / Task	Update Privacy	Integrity / Auditability	Edge-Oriented
[33]	IoT FL	Secure aggregation in TEE	Blockchain	Yes
[55]	Healthcare FL	SMPC	Blockchain + model verification	—
[22]	General FL	Secure aggregation	Backdoor detectability	—
[23]	FL networks	Secure aggregation	Verifiable aggregate	—
[24]	Edge FL	Secure aggregation	Verifiable aggregate	Yes

Ref.	Domain / Task	Update Privacy	Integrity / Auditability	Edge-Oriented
[52]	IoMT intrusion detection	Secure aggregation	Reputation-aware	Yes
[54]	Clinical EEG	Secure aggregation	—	—
[35]	IoMT analytics	FHE	zk-SNARK	Yes
[36]	5G edge-IoT anomaly detection	Zero-knowledge proofs	Verifiable privacy	Yes
[51]	Edge IoT anomaly detection	DP + secure aggregation	—	Yes
[39]	Edge-AI anomaly detection	—	Audited FL	Yes
[31]	Edge IoT anomaly detection	—	Blockchain	Yes
[26]	EHR management	FL	Blockchain	Yes
[34]	Healthcare analytics	Clustered FL	Lightweight blockchain	—
[2]	IoMT anomaly detection	DP + secure enclaves	—	Yes
SATEL-FL	Smart-healthcare edge anomaly detection	Dropout-tolerant secure aggregation	Merkle-tree tamper-evident log	Yes

— indicates that the property is not a stated focus of the work.

Three gaps emerge from this analysis. First, schemes that protect individual updates through secure aggregation, homomorphic encryption or enclaves rarely keep a persistent, verifiable record of the training process across rounds [2], [21], [23]. Second, schemes that provide integrity mostly depend on blockchain consensus or zero-knowledge proofs, whose computation, storage and latency costs are hard to justify on constrained medical edge devices [31], [35], [36]. Third, few works combine privacy-preserving aggregation, auditable integrity and lightweight anomaly detection in a single design evaluated end-to-end on healthcare data [39], [51]. SATEL-FL addresses these gaps by binding dropout-tolerant secure aggregation to a Merkle-tree tamper-evident log and a lightweight edge anomaly detector, all implemented and evaluated in MATLAB.

III. PRELIMINARIES

A. Federated Averaging

Consider K clients, where client k holds a private dataset D_k with n_k records. In round r the server sends the global weights w^r to the clients; each client trains locally for E epochs and returns w_k^r. Federated Averaging (FedAvg) [7] forms the new global model as the sample-weighted mean

$$w^{r+1} = \Sigma_k \, n_k \cdot w_k^r \; / \; \Sigma_k \, n_k \qquad (1)$$

In plain FedAvg the server sees every w_k^r, which is exactly the information that gradient-inversion and membership-inference attacks exploit [18].

B. Secure Aggregation by Pairwise Masking

Secure aggregation lets the server compute (1) without seeing any individual w_k^r [21]. Every pair of clients (i, j) agrees on a shared seed s_{ij}, and each client also draws a private self-mask seed b_i. With a pseudorandom generator PRG, a fixed-point encoding $q(\cdot)$ and modulus M, client i uploads

$$y_i = q(n_i \cdot w_i) + PRG(b_i) + \Sigma_(j>i)\, PRG(s_{ij}) - \Sigma_(j<i)\, PRG(s_{ij}) \pmod M \qquad (2)$$

Summed over all clients, the pairwise masks cancel. If some clients drop out after the masks are fixed, their pairwise masks no longer cancel. To remove them, each client secret-shares its key material with Shamir's t-out-of-K scheme, so that any t surviving clients can help the server reconstruct the missing masks while fewer than t learn nothing [21].

C. Merkle Trees and Tamper-Evident Logs

A tamper-evident log stores entries $e_1, \ldots, e_n$ in an append-only Merkle tree [38]. Leaves are hashed as $H(0x00 \| e)$ and internal nodes as $H(0x01 \| left \| right)$, with SHA-256 as H; the distinct prefixes prevent a leaf from being passed off as an internal node. The root commits to the whole sequence. An inclusion proof that e_m is in a tree of size n contains $\lceil \log_2 n \rceil$ sibling hashes, and a consistency proof that a tree of size m is a prefix of a tree of size n contains O(log n) hashes. The log operator periodically publishes a signed tree head (STH) containing the size and root; chaining each STH to the hash of its predecessor follows the forward-integrity idea of secure audit logs [37].

IV. SYSTEM AND THREAT MODEL

A. Entities

- Edge clients: hospital or ward gateways that collect IoMT network and biometric records, train the detector locally and take part in secure aggregation.
- Aggregation server: coordinates rounds, sums masked updates, removes masks and maintains the training log.
- Witness: an independent party (for example a regulator, an ethics board or a member of a hospital consortium) that stores every STH and checks that the log only grows.
- Auditor: any party (a client, a witness or an external reviewer) that verifies the log and its proofs after training.

B. Adversary Model

- T1, curious server: follows the protocol but tries to learn individual client updates (honest-but-curious).
- T2, network adversary: observes all traffic between clients and the server.
- T3, log tamperer: an outsider or an insider with access to the server, including its signing key, who modifies, deletes, reorders, inserts or rolls back training records, or presents different histories to different parties.
- T4, malicious clients: send poisoned updates. SATEL-FL does not claim to prevent poisoning; its effect is measured in Section VIII-F.

C. Security Goals and Assumptions

The goals are (G1) confidentiality of each client's update against T1–T2 and against collusion of the server with fewer than t clients; (G2) correctness, meaning the secure aggregate equals FedAvg over the surviving clients; (G3) tolerance of client dropout as long as at least $t = \lfloor K/2 \rfloor + 1$ clients survive; (G4) tamper evidence against T3; and (G5) client verifiability, meaning each client can check that its upload was recorded. The design assumes authenticated channels, a public-key infrastructure for distributing public keys, and at least one honest witness that the server does not control. Secret shares are relayed through the server but encrypted end-to-end for their recipients, as in [21], so the server never sees them in the clear. Sybil attacks, side channels and inference from the aggregate model itself are out of scope; the last can be addressed by adding differential privacy [51].

V. THE PROPOSED SATEL-FL FRAMEWORK

A. Overview

Fig. 1 shows the architecture. Each round has three phases: (i) local training on each client; (ii) secure aggregation, after which the server holds only the weighted sum of the updates; and (iii) logging, in which the server appends records of the round to the Merkle log, signs a new tree head and sends it with a consistency proof to the witness. The global model is then broadcast for the next round.

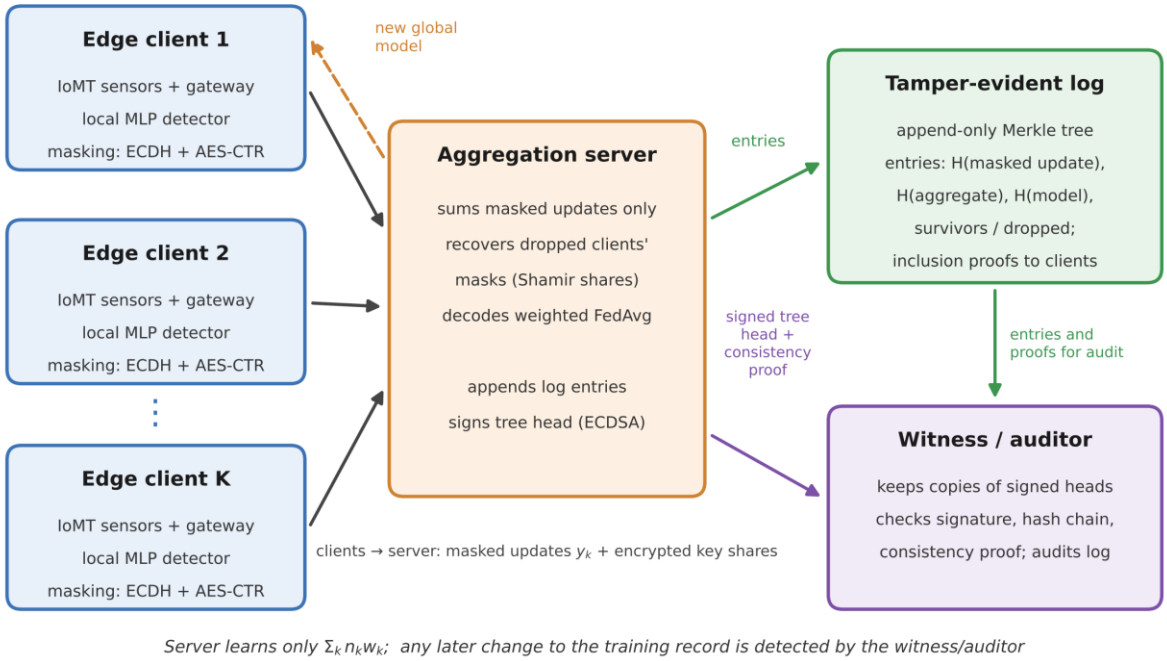

Fig. 1. SATEL-FL architecture. Clients upload only masked updates; the server records hashes of each round in a Merkle log whose signed heads are verified by an independent witness.

B. Lightweight Detector

Each client trains a multilayer perceptron with layers 35-32-16-1, ReLU activations and a sigmoid output (1,697 parameters, about 6.8 KB in single precision), minimising binary cross-entropy with Adam. A record is flagged as an attack when its predicted probability exceeds a threshold chosen to maximise F1 on the validation set. For comparison, an unsupervised autoencoder (35-16-8-16-35) trained only on normal records is also evaluated; it flags records whose reconstruction error exceeds the 99th percentile of normal validation errors.

C. Secure-Aggregation Round

Algorithm 1 lists one round. The weighted update $n_k \cdot w_k$ is encoded in fixed point with scale 2^{16} and all masking arithmetic is performed modulo $M = 2^{48}$, which is exact in 64-bit unsigned integers for up to several thousand clients. Pairwise seeds are derived as $s_{ij} = \text{SHA-256}(\text{ECDH}(i, j) \,\|\, r)$ using fresh P-256 key pairs every round, and masks are expanded with AES-128 in counter mode. Each client splits its 16-byte self-mask seed and 32-byte private key into 24 chunks of 16 bits and shares each chunk with Shamir's scheme over the prime field $p = 2^{31} - 1$ with threshold $t = \lfloor K/2 \rfloor + 1$.

Fig. 2 shows the workflow of one round, including the two decision points that distinguish SATEL-FL from plain FedAvg: recovery from client dropout and the witness check of every signed tree head.

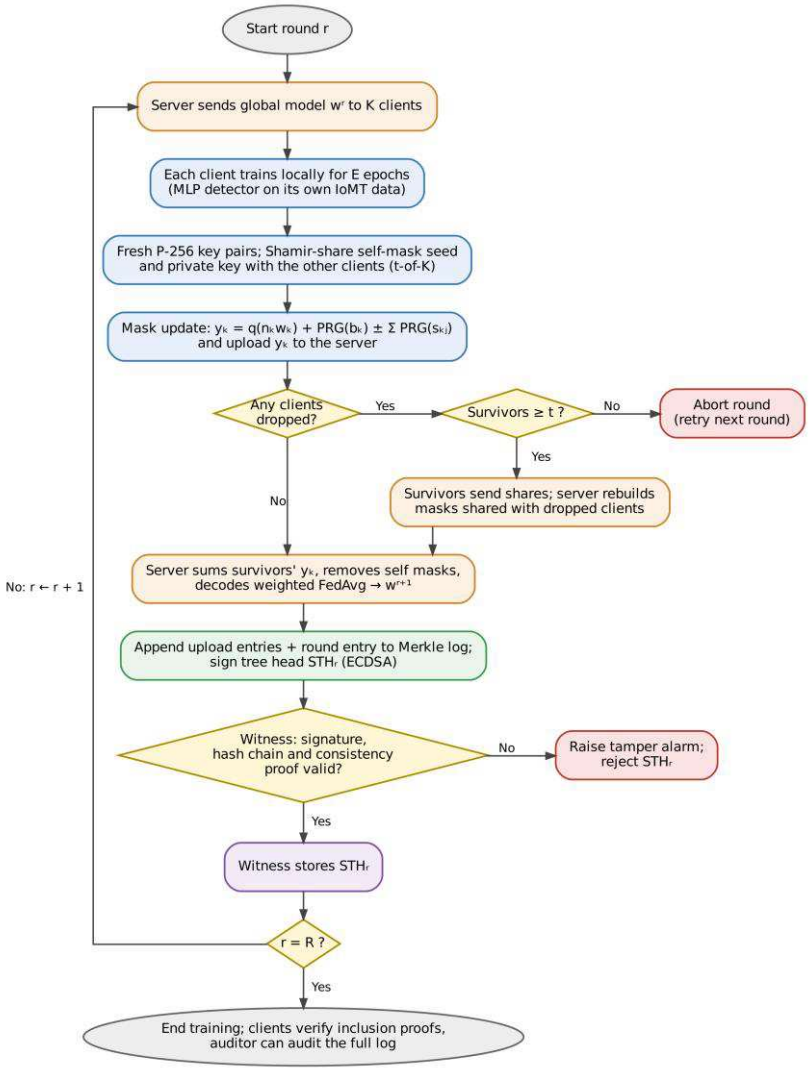

Fig. 2. Workflow of one SATEL-FL training round. Blue: client steps; orange: aggregation server; green: tamper-evident log; purple: witness; yellow: decisions.

Algorithm 1. One SATEL-FL round (round r)

1 Each client k: generate a fresh P-256 key pair; publish its public key.
2 Each client k: draw self-mask seed b_k; Shamir-share (b_k, private key) to the others (t-of-K), each share encrypted for its recipient.
3 Each client k: train locally for E epochs from $w^r \rightarrow w_k^r$.
4 Each client k: $s_{kj} \leftarrow$ SHA-256(ECDH(k, j) ‖ r) for every $j \neq k$;
upload $y_k = q(n_k \cdot w_k^r) + PRG(b_k) + \Sigma_{j>k} PRG(s_{kj}) - \Sigma_{j<k} PRG(s_{kj}) \pmod{2^{48}}$.
5 Server: $U \leftarrow$ clients whose y_k arrived; $D \leftarrow$ dropped clients; abort if $|U| < t$.
6 Survivors send their shares of b_i ($i \in U$) and of the private keys of $j \in D$.
7 Server: $S \leftarrow \Sigma_{i \in U} y_i - \Sigma_{i \in U} PRG(b_i) \mp \Sigma_{(i \in U, j \in D)} PRG(s_{ij}) \pmod{2^{48}}$;
$w^{r+1} \leftarrow$ decode(S) / $\Sigma_{(i \in U)} n_i$.
8 Server: append one upload entry per $i \in U$ and one round entry (Table II);
sign STH_r = (size, root, r, $H(STH_{r-1})$); send STH_r and a consistency proof to the witness.
9 Witness: verify the signature, the hash chain and the consistency proof; store STH_r.

A survivor's private key and a dropped client's self-mask seed are never reconstructed, so the server can never strip all masks from any single upload; this is the property that protects updates even when a client drops out after uploading [21].

D. Tamper-Evident Training Log

Table II lists the log records. Upload entries let each client later prove that its own masked update was counted; round entries bind the set of survivors and dropped clients to the resulting aggregate and model. Because only hashes of masked updates are stored, the log reveals nothing about the updates themselves.

TABLE II. Log Records Written in Each Round

Record	Fields
Upload entry (one per surviving client)	round r, client id k, sample count n_k, SHA-256 of masked update y_k
Round entry (one per round)	round r, survivors U, dropped D, $\Sigma\, n_i$, SHA-256 of aggregate S, SHA-256 of new model w^{r+1}, hash of previous STH
Signed tree head	tree size, Merkle root, round r, hash of previous STH, ECDSA (P-256) signature

E. Audit and Verification

An auditor who obtains the entries, the STHs and the witness's copy performs four checks: (a) every STH signature verifies under the server's public key; (b) every STH contains the hash of its predecessor; (c) rebuilding the Merkle tree from the entries reproduces every signed root; and (d) the latest published STH equals the witness's latest STH. A client verifies its own upload with an inclusion proof against the latest root, which requires only the leaf hash, $\lceil \log_2 n \rceil$ sibling hashes and the signed root.

F. Complexity

Per round, a client performs $K - 1$ key agreements and K mask expansions of P words, which is $O(KP)$ work, and uploads $6P$ bytes of masked weights plus a 91-byte public key and $96(K - 1)$ bytes of shares. Without dropout the server performs $O(|U| \cdot P)$ work; each dropped client adds $|U|$ key agreements and mask expansions, which is $O(|D| \cdot |U| \cdot P)$. Appending an entry costs $O(\log n)$ hashes, and inclusion and consistency proofs have $O(\log n)$ size.

VI. SECURITY ANALYSIS

Update confidentiality (G1). Each y_k is masked by $PRG(b_k)$ and by pairwise masks derived from ECDH secrets that the server cannot compute. Any mask is removable only through t shares, and for each client the honest survivors release shares of either its self-mask seed (if it survived) or its private key (if it dropped), never both. Against an honest-but-curious server colluding with fewer than t clients, the server therefore learns only $\Sigma\, n_i\, w_i$ over the survivors, following the argument of Bonawitz et al. [21].

Correctness (G2) and dropout tolerance (G3). The masks cancel exactly in modular arithmetic, so the only deviation from FedAvg is fixed-point rounding (at most 2^{-17} per weight before division). In every round of every experiment, the implementation asserts that the secure aggregate differs from plain FedAvg by less than 10^{-3}; the largest observed difference was 6.6×10^{-9} (Section VIII).

Tamper evidence (G4). Modifying, inserting, deleting or reordering any entry changes the Merkle root at every later STH, which an auditor detects by rebuilding the tree; avoiding this would require a SHA-256 collision. Changing an STH without the key breaks its ECDSA signature, and removing one breaks the hash chain. An insider who holds the signing key can rebuild and re-sign a modified history, or roll the log back, but the result no longer matches the STHs already accepted by the witness. The witness, moreover, accepts a new STH only with a valid consistency proof. Without any witness this insider attack is undetectable, which Section VIII-D confirms experimentally; SATEL-FL therefore requires at least one honest witness.

Client verifiability (G5). Each upload entry contains the hash of the client's own masked upload, so a client can confirm with an inclusion proof that its contribution was recorded, and was not silently dropped or replaced, without seeing any other client's data.

Limitations. Secure aggregation hides individual updates, so the server cannot apply Byzantine-robust rules that inspect them; Section VIII-F quantifies the consequence. In the MATLAB prototype, self-mask seeds and Shamir coefficients are drawn from MATLAB's Mersenne Twister generator for reproducibility, while key pairs, ECDH and AES come from the Java security provider. Because all clients run in one process, the prototype also passes shares directly instead of encrypting them for each recipient. A deployment must draw all secrets from a cryptographically secure generator such as SecureRandom and encrypt every share end-to-end.

VII. EXPERIMENTAL SETUP

A. Dataset and Pre-processing

WUSTL-EHMS-2020 [61], [62] contains 16,318 records collected from a real-time enhanced healthcare monitoring testbed: 14,272 normal records (87.5%) and 2,046 attacks, of which 1,124 are spoofing and 922 data-alteration attacks. Each record combines network-flow features with patient biometrics (temperature, SpO_2, pulse rate, blood pressure, heart rate, respiration rate and ST deviation). The dataset is publicly available from the WUSTL EHMS 2020 Dataset repository of Washington University in St. Louis (https://research.engineering.wustl.edu/~jain/ehms/index.html) [62]. The biometric values in it were recorded by the dataset creators on their monitoring testbed and are distributed in de-identified form; no data were collected from patients for the present study.

Identifier columns (source and destination MAC and IP addresses, ports and packet index) and the attack-category column were removed. In this dataset the identifier columns reveal the attacking machine: a random forest trained with them reached 99.98% accuracy, against 94.0% without them (Section VIII-G), so results reported with identifiers do not reflect detection ability. The Argus flag field was one-hot encoded, heavy-tailed non-negative traffic features were log-transformed, constant columns were dropped and all features were standardised using training-set statistics only, giving 35 features. The data were split 70/15/15 into training, validation and test sets, stratified by label; the test set contains 306 attacks (173 spoofing and 133 data alteration). Section VIII-H repeats the main experiments on a second dataset, CICIoMT2024.

B. Federated Partitioning

Training records were distributed among K clients either IID or non-IID. For non-IID splits, each class was divided among clients with proportions drawn from a Dirichlet distribution with concentration $\alpha \in \{1.0, 0.5, 0.1\}$; smaller α gives more skewed clients (Fig. 3). Unless stated otherwise, $K = 10$.

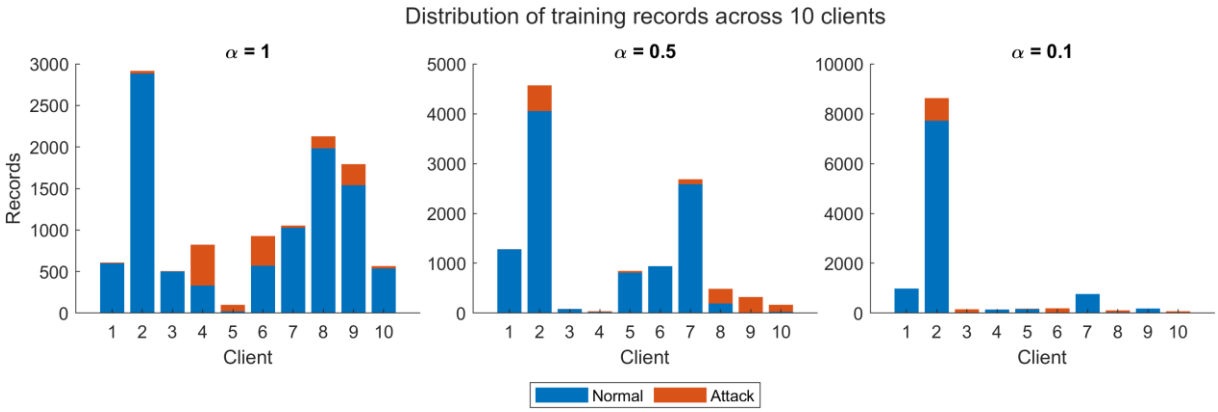

Fig. 3. Distribution of normal and attack records across 10 clients for Dirichlet concentrations α = 1.0, 0.5 and 0.1.

C. Training Configuration and Baselines

TABLE III. Hyperparameters

Parameter	Value
Detector	MLP 35-32-16-1 (ReLU, sigmoid output), 1,697 parameters
Optimiser, learning rate, batch size	Adam, 10^{-3}, 64
Rounds R, local epochs E, clients K	50, 2, 10 (10–100 in scalability tests)
Decision threshold	maximum F1 on the validation set
Masking	modulus 2^{48}, fixed-point scale 2^{16}, AES-128-CTR PRG, P-256 ECDH, SHA-256
Secret sharing	Shamir over GF($2^{31} - 1$), threshold $\lfloor K/2 \rfloor + 1$, 16-bit chunks
Log	RFC 6962-style Merkle tree, SHA-256, ECDSA P-256 signed tree heads
Repetitions	5 seeds (model initialisation and training order); fixed data split

SATEL-FL is compared with: centralised training on all data (upper bound); local-only training, where each client trains alone for the same 100 epochs (lower bound, averaged over clients); plain FedAvg; an unsupervised federated autoencoder; a permissioned blockchain with PBFT consensus as an alternative integrity layer; and coordinate-wise median and 20% trimmed-mean aggregation as poisoning defences that require plaintext updates.

D. Metrics and Implementation

Detection is reported as accuracy, precision, recall, F1-score, false-positive rate (FPR), ROC-AUC and PR-AUC (average precision), together with recall per attack type. Costs are reported as wall-clock time per round for each protocol stage, bytes uploaded per client and log size. Everything was implemented in MATLAB R2024b with the Deep Learning and Statistics and Machine Learning Toolboxes; cryptographic primitives use the Java security provider bundled with MATLAB. Experiments ran on a single PC (Intel Core i3, 8 GB RAM, Windows), with clients simulated sequentially. In the blockchain comparison, the ECDSA and SHA-256 costs are measured on this machine and the network delay is modelled as 10 ms per message hop. The CICIoMT2024 experiments of Section VIII-H do not involve the cryptographic layer and were run in Python (NumPy).

VIII. RESULTS AND DISCUSSION

A. Detection Performance

Table IV reports the mean and standard deviation over five seeds. SATEL-FL produced exactly the same predictions as plain FedAvg; its ROC-AUC and PR-AUC differed only in the fifth decimal place because of fixed-point rounding (which changes the rounded PR-AUC in Table IV by 0.001). Secure

aggregation therefore costs no accuracy. With IID clients, SATEL-FL reached an F1-score of 67.5 ± 0.9% and a ROC-AUC of 0.882 ± 0.005, compared with 71.3 ± 1.8% and 0.913 ± 0.011 for centralised training, a gap of 3.8 F1 points. It outperformed local-only training (67.0 ± 0.2%, AUC 0.858), showing that collaboration helps even though no client reveals its data. With non-IID clients ($\alpha = 0.5$) performance was slightly higher (68.8 ± 0.8%, AUC 0.886). The unsupervised autoencoder reached only 64.9% F1 and an AUC of 0.742 under FedAvg, so the supervised detector is the better choice when labelled attacks are available. Fig. 4 shows the corresponding ROC curves.

TABLE IV. Detection Performance on the Test Set (Mean ± Std over 5 Seeds, K = 10)

Method	Acc (%)	Prec (%)	Rec (%)	F1 (%)	FPR (%)	ROC-AUC	PR-AUC	Spoof rec. (%)
Centralised	93.94 ± 0.48	87.52 ± 4.44	60.26 ± 1.77	71.32 ± 1.84	1.25 ± 0.54	0.913 ± 0.011	0.776 ± 0.019	29.7 ± 3.1
Local-only (mean)	93.50 ± 0.05	91.87 ± 0.89	52.84 ± 0.28	67.03 ± 0.17	0.69 ± 0.09	0.858 ± 0.002	0.679 ± 0.013	16.6 ± 0.5
FedAvg, IID	93.53 ± 0.41	91.40 ± 6.96	53.73 ± 1.56	67.54 ± 0.89	0.78 ± 0.67	0.882 ± 0.005	0.733 ± 0.006	18.2 ± 2.8
SATEL-FL, IID	93.53 ± 0.41	91.40 ± 6.96	53.73 ± 1.56	67.54 ± 0.89	0.78 ± 0.67	0.882 ± 0.005	0.732 ± 0.006	18.2 ± 2.8
SATEL-FL, $\alpha = 0.5$	93.76 ± 0.30	92.37 ± 5.36	54.90 ± 1.60	68.78 ± 0.76	0.68 ± 0.53	0.886 ± 0.004	0.739 ± 0.004	20.2 ± 2.8
Autoencoder FedAvg, IID†	93.01	87.29	51.63	64.89	1.07	0.742	—	14.5

Data-alteration recall was 100% for every method. † Unsupervised detector, single run. Precision has a large spread because the F1-optimal threshold moves between seeds.

Recall differs sharply between attack types. Every method detected 100% of data-alteration attacks, which change the flow flags and timing, but only 16–30% of spoofing attacks, whose flows look almost identical to normal traffic once identifiers are removed. This is the main source of the moderate overall recall, and it is a property of the leakage-free task rather than of the federated setting. The convergence curves in Fig. 5 are still rising after 50 rounds, so more rounds would likely narrow the gap to centralised training further.

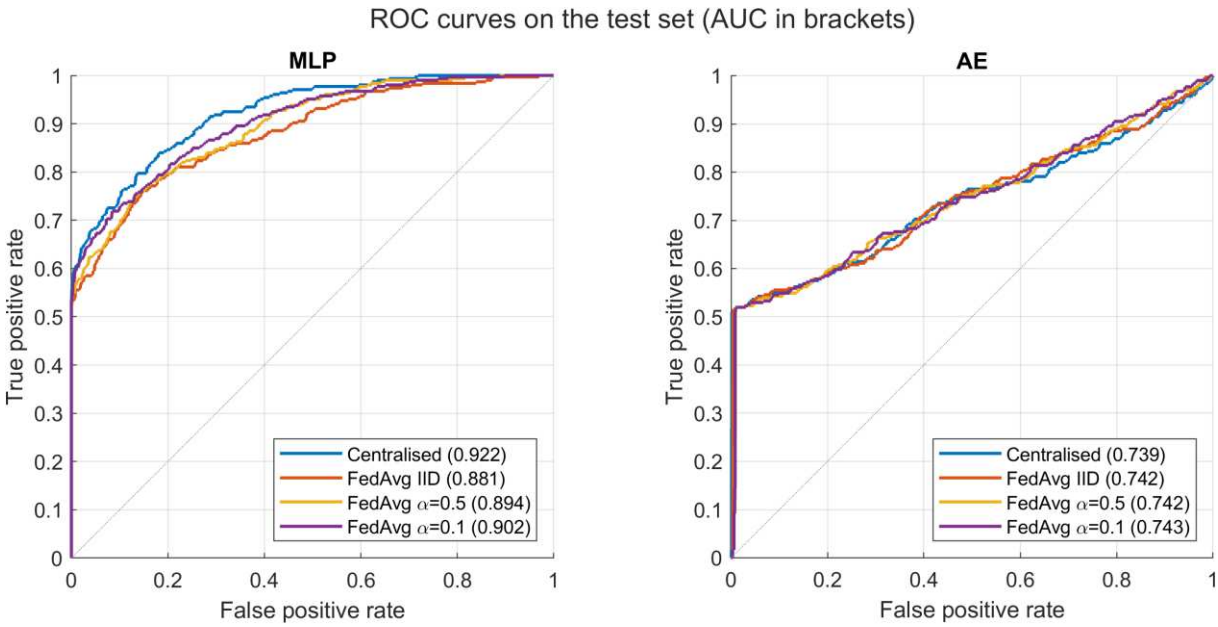

Fig. 4. ROC curves on the test set for the MLP (left) and the autoencoder (right), single seed. SATEL-FL coincides with FedAvg.

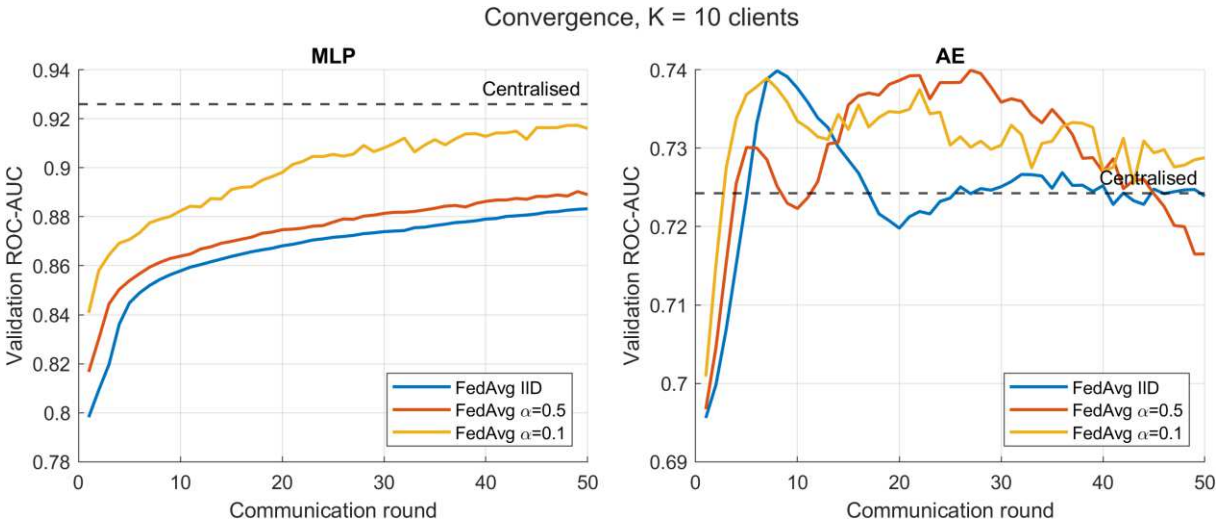

Fig. 5. Validation ROC-AUC per communication round for FedAvg with IID and non-IID clients; the dashed line is centralised training.

B. Non-IID Data and Client Dropout

Table V shows single-seed results of SATEL-FL under different data splits and dropout rates. Accuracy is essentially unaffected by dropout: with 30% of clients dropping out in every round, F1 fell by less than one point (68.4% to 67.5%) because the aggregate is simply formed over the survivors. The cost of dropout falls on the server, whose unmasking time grew from about 10 ms to 53 ms because it must rebuild the masks shared with each dropped client. In every round the secure aggregate matched plain FedAvg to within 6.6×10^{-9}.

TABLE V. SATEL-FL under Non-IID Data and Client Dropout (K = 10, Single Seed)

Data	Dropout	F1 (%)	ROC-AUC	Max |SA − FedAvg|	Client (ms/round)	Server (ms/round)
IID	0%	68.43	0.881	6.6×10^{-9}	28.7	9.8
α = 0.5	0%	68.08	0.895	5.1×10^{-9}	25.9	5.3
α = 0.1	0%	71.49	0.902	5.3×10^{-9}	24.8	5.0
IID	10%	67.87	0.880	6.2×10^{-9}	23.7	28.9
IID	20%	67.49	0.882	6.4×10^{-9}	22.9	42.4
IID	30%	67.47	0.882	6.3×10^{-9}	22.5	53.0

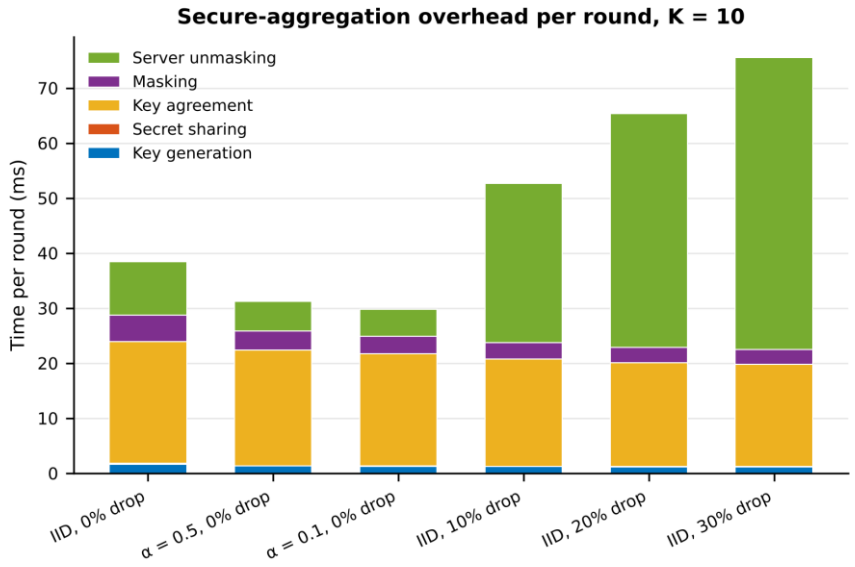

Fig. 6. Breakdown of secure-aggregation time per round (K = 10). Key agreement dominates the client side; server unmasking grows with dropout.

C. Secure-Aggregation Cost and Scalability

With 10 clients a client spends 22–29 ms per round on cryptography, about 20 ms of it on key agreement (Fig. 6, Table V), against roughly 0.5 s of local training. The masked upload is 11,137 bytes, against 6,788 bytes for plain FedAvg: the 48-bit masked weights add 50% and the public key and secret shares add the rest. Table VI and Fig. 7 show how these costs grow with the number of clients. Client cost grows with K (48 ms at K = 10, 284 ms at K = 100), because each client performs K − 1 key agreements and mask expansions; the upload grows to 19,777 bytes at K = 100 because of the shares. Logging stays between 8 and 38 ms per round. The steepest growth is in server unmasking under dropout, which reached 3.9 s per round at K = 100 with 10% dropout, since every dropped client's masks must be recomputed with every survivor. This cost, inherited from the original protocol, motivates sparse-graph variants such as SecAgg+ for larger deployments.

TABLE VI. SATEL-FL Cost per Round versus Number of Clients (5 Rounds per Setting)

K	Local training per client (ms)	Client secure aggregation (ms)	Server, 0% drop (ms)	Server, 10% drop (ms)	Logging (ms)	Upload (bytes)
10	525	47.9	19.8	81.4	14.6	11,137
20	279	53.1	13.5	98.2	8.5	12,097
50	71	141.8	62.5	736.1	12.8	14,977
100	38	283.8	208.9	3,918.3	20.3	19,777

Plain FedAvg uploads 6,788 bytes regardless of K. Absolute times vary between runs on the same machine (client cost at K = 10 measured 26–48 ms across experiments).

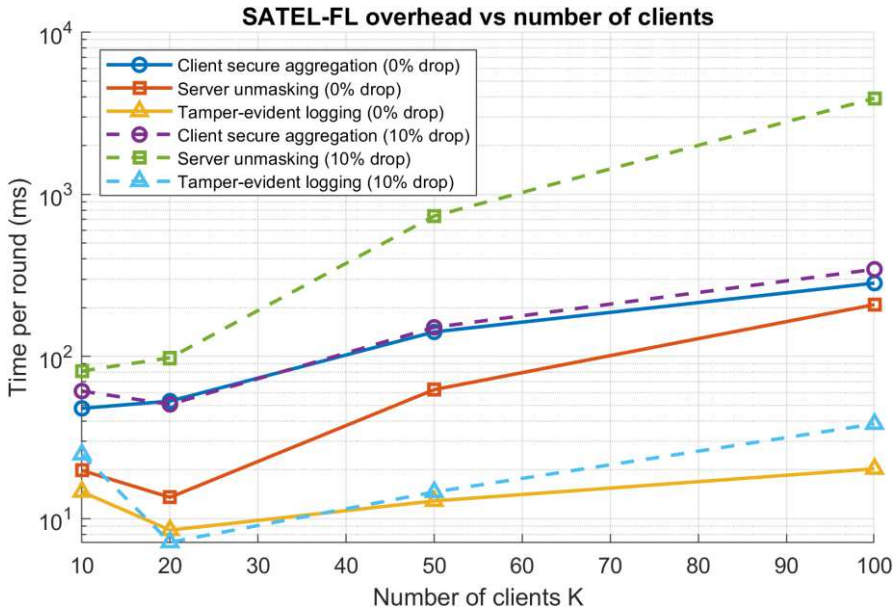

Fig. 7. Per-round cost of SATEL-FL versus the number of clients (logarithmic time axis).

D. Tamper-Evident Log: Overhead and Detection

Over 50 rounds with 10 clients the log grew to 550 entries and 101 KB, about 2 KB per round. Appending the entries, signing the tree head and having the witness verify it took 8–16 ms per round in total. A client's inclusion proof was 313 bytes and took 0.43 ms to verify, and a full independent audit of the whole training history took 0.33 s (Table VII).

TABLE VII. Logging Overhead over 50 Rounds (K = 10)

Measure	0% dropout	20% dropout
F1 (%) / ROC-AUC	68.43 / 0.881	67.49 / 0.882
Append entries (ms/round)	4.7	2.4
Sign tree head (ms/round)	5.4	2.2
Witness check (ms/round)	5.6	3.3
Total logging (ms/round)	15.7	7.9
Log entries / size	550 / 101 KB	450 / 85 KB
Inclusion proof: size / verify time	313 B / 0.43 ms	288 B / 0.41 ms
Consistency proof size	225 B	218 B
Full audit of the log	0.33 s	0.23 s

The 0% run was executed first and includes Java start-up effects, which explains its higher logging time.

Table VIII reports the tamper-detection experiments. One hundred random instances of each attack were applied to the real training log and audited. All 800 instances of the first eight attacks were detected, including single-bit modifications. The check that caught each attack matches the analysis in Section VI. Merkle-root recomputation caught entry modification, deletion, reordering and insertion; signatures and the hash chain caught forged or removed STHs; and only the witness caught rollback and the insider who re-signed a rewritten log. When the witness was removed, none of the 100 insider rewrites were detected, which confirms that the witness is a necessary part of the design.

TABLE VIII. Tamper-Detection Results (100 Random Instances per Attack)

Attack	Detected	Caught by
Modify one bit of an entry	100/100	Merkle root

Attack	Detected	Caught by
Delete an entry	100/100	Merkle root
Swap two entries	100/100	Merkle root
Insert a forged upload entry	100/100	Merkle root
Change the root inside an STH	100/100	Signature, hash chain, root
Remove an STH from the chain	100/100	Hash chain
Roll the log back to an earlier round	100/100	Witness
Insider rewrites an entry and re-signs all STHs	100/100	Witness
Same insider attack, no witness (ablation)	0/100	—

Proof sizes grow logarithmically with the log (Fig. 8): from 220 to 467 bytes for inclusion proofs and from 223 to 461 bytes for consistency proofs as the log grows from 100 to 20,000 entries. Verification time shows no growth trend: inclusion proofs verified in 1.0–1.6 ms and consistency proofs in 1.2–2.4 ms (averages over 200 and 50 proofs), and the variation between sizes is timing noise at the millisecond scale. Even 100 clients training for 200 rounds (about 20,000 entries) therefore need proofs of under half a kilobyte.

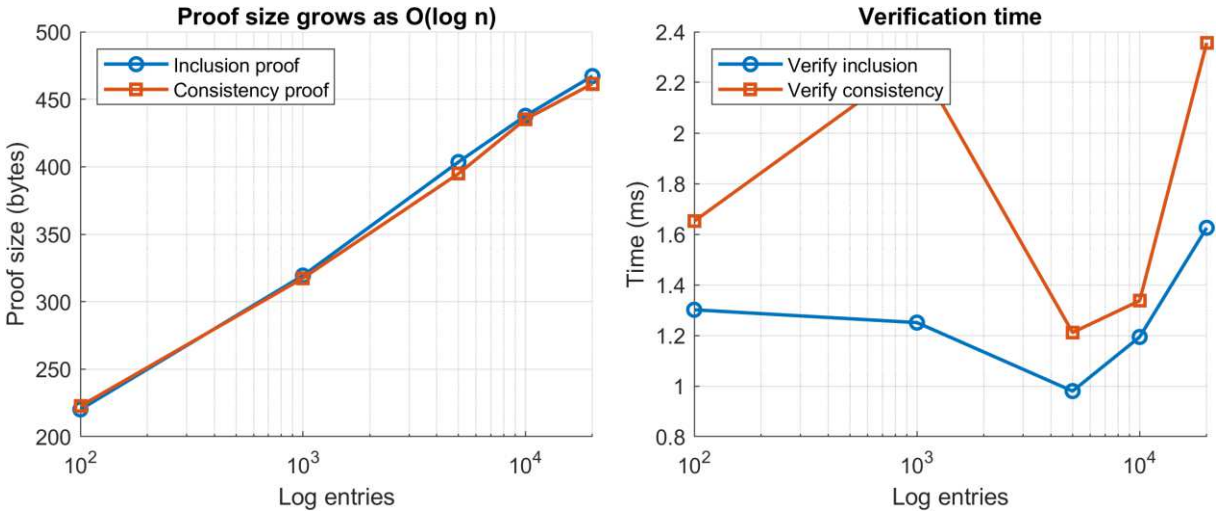

Fig. 8. Inclusion and consistency proof size (left) and verification time (right) versus log size.

E. Comparison with Blockchain Logging

Table IX compares the SATEL-FL log with recording the same per-round entries on a permissioned blockchain running PBFT. The SATEL-FL figures come from replaying the real training log; the PBFT figures combine ECDSA and SHA-256 costs measured on the same machine with the operation counts of one PBFT round, and both assume 10 ms per message hop. PBFT needs every validator to verify every transaction and the other validators' prepare and commit messages, and every validator stores a full copy of the chain. SATEL-FL needs one signature, one witness verification and one message per round. Its latency was 4.8–6.6 times lower and its storage 5.4–17.5 times lower, and its cost does not grow with the number of validators. PBFT tolerates Byzantine validators without a trusted witness; SATEL-FL trades this for a much lighter protocol that needs only one honest witness.

TABLE IX. Per-Round Cost of Integrity Layers (K = 10, 50 Rounds)

Integrity layer	Compute (ms)	Latency (ms)	Signatures	Verifications	Messages	Storage (KB)
SATEL-FL Merkle log + witness	11.6	21.6	1	1	1	108
PBFT blockchain, 4 validators	64.5	104.5	20	72	25	586
PBFT blockchain, 7 validators	83.3	123.3	26	168	85	1,173

Integrity layer	Compute (ms)	Latency (ms)	Signatures	Verifications	Messages	Storage (KB)
PBFT blockchain, 10 validators	102.0	142.0	32	300	181	1,887

Latency includes a modelled 10 ms delay per message hop (1 hop for SATEL-FL, 4 for PBFT). Storage counts all replicas.

F. Robustness to Poisoning

Table X and Fig. 9 show the effect of malicious clients. Label flipping by 10–30% of the clients reduced the F1-score of SATEL-FL only slightly (from 68.4% to 67.2%), because the honest majority dominates the average. Scaled sign-flipping model poisoning was far more damaging: with 30% malicious clients the F1-score collapsed to 20.6% and the ROC-AUC to 0.455, worse than chance. Coordinate-wise median and trimmed mean kept the F1-score above 66.5% in every case, but they must inspect each client's update in plaintext and are therefore incompatible with secure aggregation. The tamper-evident log records which clients contributed to each round, so poisoning can be investigated afterwards, but it cannot prevent it. Combining secure aggregation with robust aggregation, for example through verifiable or backdoor-detecting secure aggregation [19], [22], is the main direction for future work.

TABLE X. F1-Score (%) under Poisoning (K = 10, IID)

Attack	Malicious clients	FedAvg / SATEL-FL	Median (no privacy)	Trimmed mean (no privacy)
None	0%	68.43	68.28	68.72
Label flip	10%	68.32	68.85	68.30
Label flip	30%	67.22	68.35	68.05
Sign flip (λ = 3)	10%	66.53	68.17	68.17
Sign flip (λ = 3)	30%	20.55	67.64	66.53

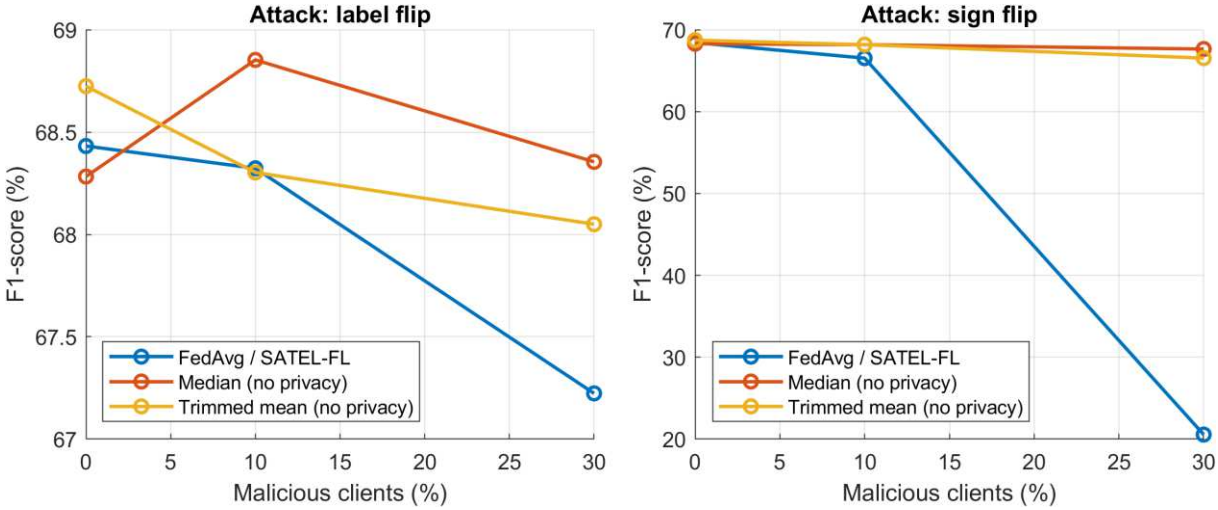

Fig. 9. F1-score versus fraction of malicious clients for label-flipping (left) and sign-flipping (right) attacks. The left panel uses a narrow F1 range (67–69%) to make small differences visible.

G. Identifier Leakage in WUSTL-EHMS-2020

To check whether published accuracies on this dataset can be compared with ours, the same random forest (100 trees) was trained on one split with and without the identifier columns. With identifiers it reached 99.98% accuracy, 99.92% F1 and a ROC-AUC of 1.000. Without them it reached 94.0% accuracy, 69.0% F1 and a ROC-AUC of 0.892, in the same range as the MLP detectors in Table IV. Results that keep identifier columns therefore measure recognition of the attacking machine rather than detection of the attack, and cannot be compared with the leakage-free results of this paper. A concurrent

leakage audit of IoT and IoMT benchmarks [42] independently reports the same effect for the source-MAC field.

H. Second Dataset: CICIoMT2024

To check that the findings do not depend on WUSTL-EHMS-2020, the detection, dropout and poisoning experiments were repeated on CICIoMT2024 [43], [63], a recent IoMT security benchmark recorded from 40 medical devices and publicly available from the CIC IoMT Dataset 2024 repository of the Canadian Institute for Cybersecurity, University of New Brunswick (https://www.unb.ca/cic/datasets/iomt-dataset-2024.html) [63]. Its Wi-Fi and MQTT part contains 17 attack classes (DDoS, DoS, reconnaissance, MQTT attacks and ARP spoofing). From the official training and test files, 350 training and 90 test records were sampled per attack class, together with 34,000 and 8,500 benign records, which gives about 15% attacks, close to the class balance of WUSTL-EHMS-2020; 15% of the training sample was held out for validation, and the training records were split among K = 10 clients. Because SATEL-FL reproduces FedAvg exactly (Section VIII-A) and its cryptographic and logging costs do not depend on the data, these runs used a Python re-implementation of FedAvg with the same detector design (here a 40-32-16-1 MLP), optimiser, schedule (50 rounds of two local epochs) and threshold rule, so the FedAvg results stand for SATEL-FL.

CICIoMT2024 has its own leakage problem. The same random forest as in Section VIII-G, trained on the inter-arrival time (IAT) feature alone, reached 97.15% accuracy and an F1-score of 90.24%, and the Number and Weight features alone each reached about 80% F1. Such single-feature separability points to a recording artifact rather than attack behaviour, so these three features were removed, together with the constant Drate and the duplicate Srate, leaving 40 features. The effect is smaller than for the WUSTL-EHMS-2020 identifiers: the random forest reached 96.88% F1 with all features and 94.29% without the three artifacts, because flooding attacks remain separable by genuine traffic features. A detector that keeps them, however, can score highly without modelling traffic at all.

Table XI reports detection performance. The ordering matches WUSTL-EHMS-2020: FedAvg, and therefore SATEL-FL, reached an F1-score of 89.4 ± 3.4% and a ROC-AUC of 0.985 ± 0.001, between local-only training (88.5 ± 1.6%, 0.977) and centralised training (90.4 ± 4.5%, 0.989). The gaps are smaller than on WUSTL-EHMS-2020 because each client already holds about 3,400 records covering every attack class, so local models are strong; collaboration still raised the ROC-AUC from 0.977 to 0.985. Non-IID clients ($\alpha = 0.5$) reached 91.6 ± 4.6% (ROC-AUC 0.985), and dropping 10% or 30% of the clients in every round left the ROC-AUC unchanged at 0.985. The F1 standard deviations (1.6–7.1 points) come mainly from the F1-optimal threshold, which moves between seeds; the threshold-free ROC-AUC varied by at most 0.003.

TABLE XI. Detection Performance on CICIoMT2024 (Mean ± Std, K = 10)

Setting	F1 (%)	ROC-AUC	Accuracy (%)	FPR (%)
Centralised	90.4 ± 4.5	0.989 ± 0.003	96.91	1.65
Local-only (mean)	88.5 ± 1.6	0.977 ± 0.001	96.31	1.75
SATEL-FL, IID	89.4 ± 3.4	0.985 ± 0.001	96.62	1.77
SATEL-FL, $\alpha = 0.5$	91.6 ± 4.6	0.985 ± 0.003	97.32	1.16
10% dropout†	89.0 ± 7.1	0.985 ± 0.002	96.34	2.15
30% dropout†	89.2 ± 6.2	0.985 ± 0.002	96.47	1.99

Five seeds, except † (three seeds; IID, with 10% or 30% of the clients dropping out in every round). SATEL-FL rows were run as FedAvg, which SATEL-FL reproduces exactly (Table IV). Local-only: each client trains alone for 100 epochs; mean over clients.

Table XII repeats the poisoning experiment. As on WUSTL-EHMS-2020, label flipping lowered the F1-score of FedAvg/SATEL-FL moderately (to 87.7% and 82.6% with 10% and 30% malicious clients), whereas scaled sign flipping by 30% of the clients collapsed it to 7.0%. Median (91.5%) and trimmed mean (86.2%) resisted, but only by inspecting plaintext updates. The need to combine secure aggregation with robust aggregation therefore holds on both datasets.

TABLE XII. F1-Score (%) under Poisoning on CICIoMT2024 (K = 10, IID, 3 Seeds)

Attack	Malicious clients	FedAvg / SATEL-FL	Median (no privacy)	Trimmed mean (no privacy)
None	0%	91.10	89.84	89.80
Label flip	10%	87.66	89.34	89.33
Label flip	30%	82.56	86.88	85.20
Sign flip ($\lambda = 3$)	10%	92.14	90.85	89.83
Sign flip ($\lambda = 3$)	30%	7.02	91.53	86.23

Median and trimmed mean need plaintext updates and are incompatible with secure aggregation. Mean of three seeds.

I. Discussion and Limitations

- The privacy and integrity mechanisms add no accuracy cost: SATEL-FL matches FedAvg exactly, and the gap to centralised training (3.8 F1 points) is the usual cost of federated training, which more rounds may reduce.
- Spoofing attacks remain hard to detect once identifiers are removed (16–30% recall for all methods). Richer temporal features or sequence models are needed.
- Server cost under dropout grows as $O(|D|\cdot|U|\cdot P)$ and reached 3.9 s per round at $K = 100$; sparse-graph secure aggregation would reduce it.
- All clients were simulated on one PC and network delays in the blockchain comparison were modelled; measurements on real edge hardware such as a Raspberry Pi or Jetson are future work.
- The CICIoMT2024 validation used a class-balanced subsample of the benchmark and a Python FedAvg in place of the MATLAB SATEL-FL implementation; the two are equivalent for detection (Table IV), but cryptographic and logging costs were measured on WUSTL-EHMS-2020 only.
- Tamper evidence relies on at least one honest witness; without it, an insider with the signing key can rewrite history undetected (Table VIII).
- Secure aggregation does not prevent model poisoning (Table X), and the aggregate model can still leak information, which differential privacy could address.

IX. CONCLUSION

This paper presented SATEL-FL, a federated learning framework for edge anomaly detection in smart healthcare that combines dropout-tolerant secure aggregation with a tamper-evident, witness-verified Merkle log. In a MATLAB evaluation on WUSTL-EHMS-2020 under a leakage-free protocol, SATEL-FL matched plain FedAvg exactly (F1 67.5 ± 0.9%, ROC-AUC 0.882 ± 0.005), stayed within 3.8 F1 points of centralised training and tolerated 30% client dropout. On CICIoMT2024, with three artifact features removed, the federated detector reached an F1-score of 89.4% and a ROC-AUC of 0.985, again between local-only and centralised training. Its cryptographic cost was tens of milliseconds per client per round for 10 clients, and its logging cost 8–16 ms per round. The log detected all 800 tampering attacks of eight types, needed proofs of under half a kilobyte for logs of 20,000 entries, and was 4.8–6.6 times faster and 5.4–17.5 times smaller than PBFT blockchain logging. The experiments also

identified the limits of the design: it needs an honest witness, its server cost grows quickly with dropout at large scale, and, on both datasets, it offers no protection against model poisoning. Future work will integrate robust secure aggregation, add differential privacy to the aggregate, use sparse-graph masking for larger federations and deploy the framework on real edge hardware.

DECLARATIONS

Funding: The authors did not receive support from any organization for the submitted work. No funding was received for conducting this study.

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

Ethics approval: Not applicable. This study did not involve human participants, human tissues or animals. It was conducted entirely on publicly available, de-identified benchmark datasets (WUSTL-EHMS-2020 [61], [62] and CICIoMT2024 [43], [63]) that were recorded in laboratory testbeds by their creators, and no new data were collected from individuals; therefore, formal ethics committee approval was not required.

Consent to participate: Not applicable. This study used publicly available, de-identified datasets and did not involve the direct participation of identifiable human participants, human tissues or animal subjects; therefore, informed consent to participate was not required.

Consent to publish: Not applicable. The manuscript contains no data, images or other material from which any individual could be identified. Only publicly available, de-identified datasets were used, with no identifiable human participants, human tissues or animal subjects; therefore, consent for publication was not required.

Clinical trial number: not applicable. This study is not a clinical trial.

Data availability: The datasets analysed during the current study are available in the following public repositories: WUSTL-EHMS-2020 in the WUSTL EHMS 2020 Dataset repository of Washington University in St. Louis, https://research.engineering.wustl.edu/~jain/ehms/index.html [62]; and CICIoMT2024 in the CIC IoMT Dataset 2024 repository of the Canadian Institute for Cybersecurity, University of New Brunswick, https://www.unb.ca/cic/datasets/iomt-dataset-2024.html [63]. No new data were collected from human participants in this study.

Code availability: The MATLAB scripts that reproduce all results are listed in the Appendix and are available from the corresponding author on reasonable request.

REFERENCES

[1] A. Rahman et al., "Federated learning-based AI approaches in smart healthcare: Concepts, taxonomies, challenges and open issues," Cluster Computing, pp. 1–41, 2022, doi: 10.1007/s10586-022-03658-4.

[2] K. Mahmood, S. Khan, M. Abdelhaq, M. U. Hassan, M. Uddin, R. Alsaqour, K. A. Awan, and M. A. Alsoufi, "Adaptive resource aware and privacy preserving federated edge learning framework for real time internet of medical things applications," Scientific Reports, vol. 15, Art. no. 36468, 2025, doi: 10.1038/s41598-025-23398-w.

[3] S. M. Rajagopal, M. Supriya, and R. Buyya, "FedSDM: Federated learning based smart decision making module for ECG data in IoT integrated edge-fog-cloud computing environments," Internet of Things, vol. 22, Art. no. 100784, 2023, doi: 10.1016/j.iot.2023.100784.

[4] M. Shokri, "Privacy-preserving edge federated learning for intelligent mobile-health systems," Future Generation Computer Systems, vol. 161, pp. 625–637, 2024, doi: 10.1016/j.future.2024.07.035.

[5] P. Karpagam, M. Karthikeyan, G. Kalpana, and A. Suresh, "Edge-AI enabled secure IoT framework for real-time patient monitoring and anomaly detection in smart healthcare systems," Scientific Reports, vol. 16, 2026, doi: 10.1038/s41598-026-59602-8.

[6] M. Prabha, S. Nandhini, M. Dayanidhy, and R. Pradeep, "Edge-AI integrated secure wireless IoT architecture for real time healthcare monitoring and federated anomaly detection," Scientific Reports, vol. 16, 2025, doi: 10.1038/s41598-025-30150-x.

[7] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. 20th Int. Conf. Artificial Intelligence and Statistics (AISTATS), PMLR vol. 54, 2017, pp. 1273–1282.

[8] M. Akter, N. Moustafa, and B. P. Turnbull, "SPEI-FL: Serverless privacy edge intelligence-enabled federated learning in smart healthcare systems," Cognitive Computation, vol. 16, pp. 2626–2641, 2024, doi: 10.1007/s12559-024-10310-3.

[9] A. Raza, K. Tran, L. Koehl, and S. Li, "AnoFed: Adaptive anomaly detection for digital health using transformer-based federated learning and support vector data description," Engineering Applications of Artificial Intelligence, vol. 121, Art. no. 106051, 2023, doi: 10.1016/j.engappai.2023.106051.

[10] D. Gupta, O. Kayode, S. Bhatt, M. Gupta, and A. Tosun, "Hierarchical federated learning based anomaly detection using digital twins for smart healthcare," in Proc. IEEE 7th Int. Conf. Collaboration and Internet Computing (CIC), 2021, pp. 16–25, doi: 10.1109/CIC52973.2021.00013.

[11] H. Xiang, X. Zhang, X.-L. Xu, A. Beheshti, L.-Y. Qi, Y. Hong, and W.-C. Dou, "Federated learning-based anomaly detection with isolation forest in the IoT-edge continuum," ACM Transactions on Multimedia Computing, Communications and Applications, vol. 22, pp. 1–19, 2024, doi: 10.1145/3702995.

[12] R. Raj, B. L. Pal, and S. Singh, "Federated learning for privacy-preserving anomaly detection in heterogeneous IoT networks," International Journal of Computer Information Systems and Industrial Management Applications, 2026, doi: 10.70917/ijcisim-2026-4306.

[13] H. Al-Alshaikh, "A federated learning–enabled energy-aware anomaly detection algorithm for secure big data analytics in IoT-based smart healthcare systems," Scientific Reports, vol. 16, 2026, doi: 10.1038/s41598-026-53494-4.

[14] K.-J. Zhang, X.-Y. Yu, J. Wang, W.-B. Wang, M. Li, Y. Yan, and Y. Zhang, "FAPAD: A federated aggregation optimization-based privacy-preserving anomaly detection framework for time series," Cybersecurity, vol. 9, 2026, doi: 10.1186/s42400-025-00511-0.

[15] J. Jithish, B. Alangot, N. Mahalingam, and K. Yeo, "Distributed anomaly detection in smart grids: A federated learning-based approach," IEEE Access, vol. 11, pp. 7157–7179, 2023, doi: 10.1109/ACCESS.2023.3237554.

[16] M. M. Mahdi, M. A. G. Raju, K.-C. Lee, and D. Kim, "Federated learning-enabled secure multi-modal anomaly detection for wire arc additive manufacturing," Machines, vol. 13, 2025, doi: 10.3390/machines13111063.

[17] I. A. Khan, I. Razzak, D.-C. Pi, U. Zia, S. Kamal, and Y. Hussain, "A novel collaborative SRU network with dynamic behaviour aggregation, reduced communication overhead and explainable features," IEEE Journal of Biomedical and Health Informatics, vol. 28, pp. 3228–3235, 2024, doi: 10.1109/JBHI.2024.3352013.

[18] S. S. Alhawas and M. A. Rassam, "Federated learning in edge computing: Vulnerabilities, attacks, and defenses—A survey," Sensors, vol. 26, 2026, doi: 10.3390/s26041275.

[19] J.-K. Liu et al., "PnA: Robust aggregation against poisoning attacks to federated learning for edge intelligence," ACM Transactions on Sensor Networks, 2024, doi: 10.1145/3669902.

[20] E. Alruwaili and T. Moulahi, "A robust and verifiable federated learning framework for preventing data poisonous threats in e-health," Frontiers in Public Health, vol. 14, 2026, doi: 10.3389/fpubh.2026.1762346.

[21] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS), 2017, pp. 1175–1191, doi: 10.1145/3133956.3133982.

[22] Z. Zhang, J. Li, S. Yu, and C. Makaya, "SAFELearning: Secure aggregation in federated learning with backdoor detectability," IEEE Transactions on Information Forensics and Security, vol. 18, pp. 3289–3304, 2023, doi: 10.1109/TIFS.2023.3280032.

[23] W. Yao, T. Zhou, Y. Han, and X. Wang, "Verifiable secure aggregation scheme for privacy protection in federated learning networks," Discover Computing, vol. 28, 2025, doi: 10.1007/s10791-025-09676-1.

[24] S. Zhang, F. Ren, W. Liang, K.-C. Li, and A.-S. K. Pathan, "VSAF: Verifiable and secure aggregation scheme for federated learning in edge computing," Computer Science and Information Systems, vol. 22, pp. 1405–1432, 2025, doi: 10.2298/CSIS250221057Z.

[25] Y.-R. Chang, C. Fang, and W. Sun, "A blockchain-based federated learning method for smart healthcare," Computational Intelligence and Neuroscience, vol. 2021, 2021, doi: 10.1155/2021/4376418.

[26] M. S and J. R., "Blockchain-enabled federated learning with edge analytics for secure and efficient electronic health records management," Scientific Reports, vol. 15, 2025, doi: 10.1038/s41598-025-12225-x.

[27] A. Nawaz, M. Irfan, X.-J. Yu, H. Aldawsari, R. Alsisi, Z. Zou, and T. Westerlund, "Blockchain-enabled privacy-preserving second-order federated edge learning in personalized healthcare," IEEE Transactions on Consumer Electronics, vol. 71, pp. 9983–9992, 2025, doi: 10.1109/TCE.2025.3620115.

[28] A. Baihan, N. Kryvinska, M. Amoon, W. Jiang, Z. Ullah, and M. Shafiq, "Cloud assisted blockchain-enabled split federated learning framework for security and privacy-preserving of IoMT in healthcare 5.0," Scientific Reports, vol. 16, 2026, doi: 10.1038/s41598-026-41771-1.

[29] M. Babar, Z. Khan, S. Kaleem, B. Qureshi, and W. Boulila, "TAWB-SFL: Trust-aware blockchain-orchestrated split federated learning for intrusion detection in 6G healthcare IoT," IEEE Open Journal of the Communications Society, vol. 7, pp. 2270–2286, 2026, doi: 10.1109/OJCOMS.2026.3667693.

[30] R. Sonani, R. Alhejaili, P. Chatterjee, K. H. Alnafisah, and J. Ali, "Secure malicious node detection in decentralized healthcare networks using cloud and edge computing with blockchain-enabled federated learning," Computer Modeling in Engineering & Sciences, 2025, doi: 10.32604/cmes.2025.070225.

[31] U. Lilhore, Y. K. Sharma, S. Pradeep, S. Rubaiee, R. Alroobaea, A. M. Baqasah, M. Alsafyani, and L. G. Tekeste, "Blockchain-enabled federated learning framework with hybrid CNN-LSTM anomaly detection for secure edge IoT networks," Journal of Cloud Computing, vol. 15, 2026, doi: 10.1186/s13677-026-00907-4.

[32] R. Huang and J. Ren, "Distributed anomaly detection and protection mechanism integrating blockchain and federated learning," in Proc. 9th Int. Symp. Big Data and Applied Statistics (ISBDAS), 2026, pp. 548–552, doi: 10.1109/ISBDAS69350.2026.11484155.

[33] A. P. Kalapaaking, I. Khalil, M. S. Rahman, M. Atiquzzaman, X. Yi, and M. Almashor, "Blockchain-based federated learning with secure aggregation in trusted execution environment for Internet-of-Things," IEEE Transactions on Industrial Informatics, vol. 19, pp. 1703–1714, 2023, doi: 10.1109/TII.2022.3170348.

[34] A. Melhem, A. Aleroud, A. Al-Mamun, M. I. Ibrahem, and N. Nasser, "Secure and resilient clustered federated learning for web-enabled healthcare analytics using lightweight blockchain and adaptive model selection," ACM Transactions on the Web, 2026, doi: 10.1145/3801730.

[35] S. B. Othman and N. Mihret, "Scalable privacy-preserving data analytics for IoMT via FHE and zk-SNARK-enabled edge aggregation," Scientific Reports, vol. 16, 2026, doi: 10.1038/s41598-026-35284-0.

[36] M. J. C. S. Reis, "ZK-FLGuard: Verifiable privacy via zero-knowledge proofs in federated anomaly detection for 5G edge-IoT systems," International Journal of Computer Theory and Engineering, vol. 18, 2026, doi: 10.7763/IJCTE.2026.V18.1386.

[37] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," ACM Transactions on Information and System Security, vol. 2, no. 2, pp. 159–176, 1999.

[38] S. A. Crosby and D. S. Wallach, "Efficient data structures for tamper-evident logging," in Proc. 18th USENIX Security Symposium, 2009, pp. 317–334.

[39] H. Azzan and A. Alkholidi, "SEAF-FL-MLP: Audited federated edge-AI anomaly detection with client heterogeneity and shortcut-sensitivity analysis," Sana'a University Journal of Applied Sciences and Technology, vol. 4, no. 7, 2026, doi: 10.59628/jast.v4i7.3475.

[40] S. Basudan and A. Alamer, "Verifiable and privacy-preserving peer-to-peer federated learning with collaborative incentives for edge-IoMT," Peer-to-Peer Networking and Applications, vol. 19, 2026, doi: 10.1007/s12083-026-02258-y.

[41] A. Rizzini, M. Esposito, T. Gagliardoni, and F. Bruschi, "SoK: Verifiable integrity claims for privacy-preserving federated learning," Proceedings on Privacy Enhancing Technologies, vol. 2026, no. 4, 2026, doi: 10.56553/popets-2026-0119.

[42] A. Mohammad and H. Mohammad, "Labelling artifacts in IoT and IoMT intrusion-detection benchmarks: A seven-benchmark leakage audit and a leakage-safe re-benchmark," SSRN preprint, Oct. 2026, doi: 10.2139/ssrn.7570408.

[43] S. Dadkhah, E. C. P. Neto, R. Ferreira, R. C. Molokwu, S. Sadeghi, and A. A. Ghorbani, "CICIoMT2024: A benchmark dataset for multi-protocol security assessment in IoMT," Internet of Things, vol. 28, Art. no. 101351, 2024, doi: 10.1016/j.iot.2024.101351.

[44] A. Maurya, R. Haripriya, M. Pandey, J. Choudhary, D. Singh, S. Solanki, and D. Sharma, "Federated learning for privacy-preserving severity classification in healthcare: A secure edge-aggregated approach," IEEE Access, vol. 13, pp. 102339–102358, 2025, doi: 10.1109/ACCESS.2025.3576135.

[45] W. Liu, Y.-H. Zhang, G. Han, J. Cao, H. Cui, and D. Zheng, "Secure and efficient smart healthcare system based on federated learning," International Journal of Intelligent Systems, vol. 2023, pp. 1–12, 2023, doi: 10.1155/2023/8017489.

[46] H. Khan, R. Kavati, S. S. Pulkaram, and A. Jalooli, "End-to-end privacy-aware federated learning for wearable health devices via encrypted aggregation in programmable networks," Sensors, vol. 25, 2025, doi: 10.3390/s25227023.

[47] D. Xu, H.-L. Chen, Y. Zeng, Y. Yang, J. Huang, J. Song, and Y. Zhan, "Privacy-preserving federated deep learning for robust anomaly detection in distributed security sensing systems," Sensors, vol. 26, 2026, doi: 10.3390/s26123901.

[48] R. Bensaid, N. Labraoui, A. A. A. Ari, H. Saidi, and L. A. Maglaras, "SA-FLIDS: Secure and authenticated federated learning-based intelligent network intrusion detection system for smart healthcare," PeerJ Computer Science, vol. 10, 2024, doi: 10.7717/peerj-cs.2414.

[49] H. Sharma, A. Kumar, and G. Kumar, "Privacy-enhanced federated learning framework for intrusion detection in smart IoT environments," Revolutionary Advances in Computing and Electronics: An International Journal, vol. 1, no. 2, 2025, doi: 10.65890/race.v1i2.153.

[50] S. Sengan and C.-S. Shieh, "Non-IID and aware federated intrusion detection with PBFT with secured model aggregation for multi institutional healthcare internet of things networks," Scientific Reports, vol. 16, 2026, doi: 10.1038/s41598-026-56167-4.

[51] R. R. Tang, "RedPO-BRNNet: Federated anomaly detection with differential privacy and secure aggregation for edge IoT," Informatica, vol. 49, 2025, doi: 10.31449/inf.v49i35.11175.

[52] F. F. Al-Mashakbah, A. Alqammaz, A. Khalifeh, E. F. A. Mashagbah, and E. S. Hanandeh, "Federated defense in the medical edge: A reputation-aware secure aggregation protocol for IoMT intrusion detection," International Journal of Advanced Computer Science and Applications, 2026, doi: 10.14569/ijacsa.2026.0170870.

[53] C.-H. Liu and Z.-Y. Wu, "Quantum-resistant secure aggregation for healthcare federated learning," Computers, Materials & Continua, 2026, doi: 10.32604/cmc.2026.075495.

[54] P. Rajabi and M. Toorani, "Secure aggregation for privacy-preserving federated learning on clinical EEG data," arXiv preprint arXiv:2607.28191, 2026, doi: 10.48550/arXiv.2607.28191.

[55] A. P. Kalapaaking, I. Khalil, and X. Yi, "Blockchain-based federated learning with SMPC model verification against poisoning attack for healthcare systems," IEEE Transactions on Emerging Topics in Computing, vol. 12, pp. 269–280, 2023, doi: 10.1109/TETC.2023.3268186.

[56] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 911–926, 2020, doi: 10.1109/TIFS.2019.2929409.

[57] X. Guo, Z. Liu, J. Li, J. Gao, B. Hou, C. Dong, and T. Baker, "VeriFL: Communication-efficient and fast verifiable aggregation for federated learning," IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1736–1751, 2021, doi: 10.1109/TIFS.2020.3043139.

[58] A. P. Kalapaaking, I. Khalil, and M. Atiquzzaman, "Smart policy control for securing federated learning management system," IEEE Transactions on Network and Service Management, vol. 20, pp. 1600–1611, 2023, doi: 10.1109/TNSM.2023.3276594.

[59] M. P. Rajagopal and G. Natesan, "PEF-CAPD: A privacy enhanced federated cyber physical and attack detection framework for edge-cloud-blockchain enabled smart healthcare environment," Transactions on Emerging Telecommunications Technologies, vol. 36, 2025, doi: 10.1002/ett.70187.

[60] H. Desai, M. S. Ozdayi, and M. Kantarcioglu, "BlockFLA: Accountable federated learning via hybrid blockchain architecture," in Proc. 11th ACM Conf. Data and Application Security and Privacy (CODASPY), 2021, pp. 101–112, doi: 10.1145/3422337.3447837.

[61] A. A. Hady, A. Ghubaish, T. Salman, D. Unal, and R. Jain, "Intrusion detection system for healthcare systems using medical and network data: A comparison study," IEEE Access, vol. 8, pp. 106576–106584, 2020, doi: 10.1109/ACCESS.2020.3000421.

[62] A. A. Hady, A. Ghubaish, T. Salman, D. Unal, and R. Jain, "WUSTL EHMS 2020 dataset for Internet of Medical Things (IoMT) cybersecurity research," Washington University in St. Louis, 2020. [Dataset]. Available: https://research.engineering.wustl.edu/~jain/ehms/index.html

[63] Canadian Institute for Cybersecurity, University of New Brunswick, "CICIoMT2024: Attack vectors in healthcare devices – a multi-protocol dataset for assessing IoMT device security," 2024. [Dataset]. Available: https://www.unb.ca/cic/datasets/iomt-dataset-2024.html

APPENDIX: REPRODUCIBILITY (MATLAB SCRIPTS)

All results can be regenerated by running the scripts below in order from the SATEL-FL folder. Settings are in config.m. The CICIoMT2024 experiments of Section VIII-H are reproduced by the Python scripts prep_cic.py and exp_cic.py supplied with the code.

Script	Produces	Used in
step1_prepare_data	pre-processed data, client partitions, Fig. 3	Sec. VII-A, VII-B
step2_baselines	centralised, local-only, FedAvg; MLP and autoencoder; Figs. 4–5	Table IV, Sec. VIII-A
step3_secure_aggregation	secure aggregation, non-IID and dropout runs; Fig. 6	Table V
step4_tamper_evident_log	full SATEL-FL with log, audit and inclusion proofs	Table VII
step5_tamper_detection	nine tampering attacks; proof scalability; Fig. 8	Table VIII, Fig. 8
step6_scalability_blockchain	K = 10–100 costs; PBFT comparison; Fig. 7	Tables VI, IX
step7_poisoning	label- and sign-flipping attacks; Fig. 9	Table X
step8_repeated_seeds	five-seed mean ± std	Table IV
step9_leakage_check	random forest with and without identifiers	Sec. VIII-G