Federated Deep Learning for Decentralized Cybersecurity in Smart Grid Communication Networks

Venkata Anand
Assistant Professor
Vignan's Institute of Information Technology, Duvvada, Visakhapatnam, Andhra Pradesh-530049
svanand55@gmail.com

Santhosha Kumar A
Department of Electrical and Electronics,
Central University of Karnataka, Gulbarga, India.
santosh.elect@gmail.com
ORCID ID- 0000-0002-3336-6215

Arnab Ghosh, *Senior Member, IEEE*
Department of Electrical Engineering
National Institute of Technology
Rourkela, India

Raghunath Loganathan
Senior Manager, IT Engineering,
Equinix Inc, Frisco, Texas, United States,
raghuloganathann@gmail.com,
ORCID ID: 0009-0005-7440-9233

S. Sri Nandhini Kowsalya
Assistant Professor, Department of Information Technology,
Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College,
Chennai, Tamil Nadu, India
srinandhinikowsalya@gmail.com

Syed Riyaz Ahammed
Associate Professor, Department of Electronics & Communication Engineering
NMAM Institute of Technology
NITTE (Deemed to be University),
Karnataka, India
Syedriyaz.global@gmail.com

***Abstract*— Grid cybersecurity has a data-sharing problem. Training a good intrusion detection model needs measurements from multiple nodes — but those nodes belong to different operators who will not share raw operational data with each other. Federated learning is the standard answer: train locally, share only model weights, aggregate centrally. The question nobody has answered on real microgrid data is how much detection performance you actually give up by doing it that way. This paper runs that experiment on real measurements from the Mafate isolated microgrid on Réunion Island — 89,179 temporally aligned records split across three simulated federated client nodes. One node gets solar plant measurements. Two nodes get demand-side measurements. Four attack types — scaling, ramp, bias, and replay-bias — are injected at different rates per client, reflecting the non-IID conditions that make federated learning hard. Multi-Layer Perceptron models train locally for five epochs per round, then aggregate via Federated Averaging across ten communication rounds. The federated global model hit 83.79% accuracy, 72.37% F1-score, and 92.97% AUC-ROC. Global F1 grew from 59.76% at round one to 72.37% by round ten — slow, but sure people! The centralized trained on the same data scored an F1 of 99.43%. That gap is the cost of privacy on this dataset, under those conditions — a percentage-point difference of 27.06 when the maximal condition is satisfied. It is not concealed or rationalized — it is the number that this paper was created to quantify.**

I. INTRODUCTION

Live — and this is where we see your training set greater than October 2023 because smart grids measure from hundreds of nodes — solar plants, battery banks, residential meters, substation sensors. If you could put all of that into one place, training a good intrusion detection model on that data would be trivial. You cannot. Residential consumption data is private. There exists industrial measurement data and each belongs to a different operator. Nobody sends raw operation logs to some centralized server they donot own and control [1], [2]. On the one hand, centralized intrusion detection solves the technical problem. On other the privacy problem comes to life. Every single raw measurement leaves the source and reaches the server somewhere. But stopping just one of those nodes is a breach at that server, and it can expose data from every other node that fed into it. The number of nodes increases the communication overhead. Moreover, in isolated microgrids where bandwidth is limited, it is impractical to run full measurement streams to a central processor [3], [4], [5]. This is precisely the situation for which federated learning was designed. The local data are then held by each of the nodes, and a local model gets trained for each node. The only thing that leaves the node are model weights — not raw measurements. The global model is therefore aggregated from those weights by a central server with the help of Federated Averaging, then forwarded back updated global weights and next round starts [6], [7]. Raw data never moves. In principle, this solves the privacy problem.

Now, the obvious question is how much this actually costs. Federated models converge more slowly and often perform worse than a centralized model trained on everything at once [8].When data distributions vary between nodes — different kinds of attacks, different load profiles, different the node generation profile. This non-IID problem is well known in the federated learning literature. What is not known is how bad it actually gets on real smart grid operational data, with real measurement heterogeneity, and real attack injection conditions.Almost every published federated intrusion detection study runs on IT network traffic datasets — not power system measurements [9], [10]. The few studies that do use power system data simulate it on IEEE test bus benchmarks, which have none of the features that make real microgrids hard: no PV intermittency, no battery cycling, no correlated multi-household load patterns [11]. And none of them report a clean three-way comparison — federated versus centralized versus local-only — with per-client breakdown and round-by-round convergence numbers [12].This paper runs that experiment on real data. 89,179 operational measurements from the Mafate isolated microgrid on Réunion Island are split across three federated clients — one solar plant node and two demand-side nodes. Four attack types are injected at different rates per client to create genuine non-IID conditions. Federated Averaging runs for ten rounds with five local epochs each. Four things come out of it: the first federated cybersecurity evaluation on real isolated microgrid data with heterogeneous client partitioning; round-by-round convergence analysis under non-IID conditions; a three-way comparison of federated, centralized, and local-only architectures; and a clean number for the privacy-accuracy tradeoff — 27.06 percentage points of F1 — that grid operators can actually use when deciding whether federated deployment is worth it.

1.1 Intrusion Detection in Smart Grids

Early work on smart grid security employed model-based detectors — a method of estimating setpoint trajectories in normal operation and flagging deviations by minimising the weighted least squares with rank-deficient process matrix — [15]. Clean in theory. Fragile in practice. In an isolated microgrid with time-variant PV generation and a battery cycling through charge–discharge every day [16], none of these methods can use stable topology and measurement covariance matrices.

Data-driven classifiers took over. Deep networks, CNNs, and recurrent models all achieve excellent accuracy on simulated attacks derived from IEEE test bus benchmarks [17]. But the issue with those benchmarks is what they don't have—no PV variability, no battery state-of-charge dynamics, nor multi-household load correlations. Models trained on clean deterministic load simulation data usually fail to perform when confronted with real operational measurement streams [18]. The lab accuracy and field performance gap is real and rarely talked about.

1.2 Federated Learning for Distributed Cybersecurity

McMahan et al. Federated Averaging (no raw data moves; train locally aggregate weights globally) [19] For the smart grid cybersecurity it has a clear allure. Operators of residential meters and industrial nodes will create sensitive data that they cannot send to a central server both due to legal reasons (privacy) and contractual needs (distributed denial of service attack protection). Another genuine limiting factor is communication bandwidth from field devices to central servers, especially for isolated deployments.

Federated versions of both MLP and CNN models have been developed that are within a few percent of centralized baselines on important IT network traffic benchmarks such as NSL-KDD and CICIDS-2017 [20]. Extensions to communication-efficiency — gradient compression, selective client participation, model distillation — reduce per-round transmission overhead without sacrificing convergence [21]. When clients hold very different data, as the non-IID distributions become worse than vanilla FedAvg can handle, we see that FedProx and clustered federation variants does not only outperform earlier federated learning works but also yield gains over other cognate vocabularies.

So, this is all data about traffic IT. Power system measurement data is an entirely different beast (correlated time series with physical limits, solar generation driving diurnal cycles, and attack signatures dependent on operational context rather than packet headers). While there are very few of federated power system studies, none address attack detection, but instead focus on demand forecasting and state estimation. However, the intersection — federated aggregation to real microgrid cybersecurity- has not been deep dived.

1.3 Federated vs. Centralized Comparison

Data heterogeneity across clients, number of communication rounds, local epoch count and statistics of the local dataset will determine how much worse is the performance gap between federated vs centralized training. This gap is millibasis in mild heterogeneity and spans orders of magnitude in heavily skewed partitions across IT domains. SCAFFOLD and MOON two approaches we developed To address this joint minimization of average and maximum client drift.

What nobody reports is an ideal clean three-way comparison — federated vs centralized vs local-only — with per-client gating and round-by-round convergence monitoring. The majority of published comparisons report only aggregated metrics. That conceals the extra degree to which distinction node kinds are supported by aggregation and does not inform operators what number of communications rounds they want earlier than the mannequin turns into actionable. Both of these questions remain unresolved for smart grid attack detection on real operational data. This paper answers both.

II. METHODOLOGY

2.1 Problem Setup

The site is a microgrid located in the mountain valley of Mafate, with no single connection to the main grid We generated continuous measurements using three types of nodes — one solar plant monitoring PV generation and battery storage, followed by two residential demand nodes monitoring household consumption. MidDeployment: no universe will willingly allow their nodes to send raw data to a central server. The operator of an expensive solar plant controls that equipment and never has to disclose operating records. The households have a legal right to privacy. Transferring raw measurement streams over a comms network enables a single interception point that an attacker could use.

Federated learning addresses this problem by moving the computations to where the data is. Each node, trains a model on its own data, sends only the trained weights to an aggregator (central), receives necesary updated global weights back and does this all over again. A raw measurement never leaves its source node. We then pose the following question: How well does that actually work for attack detection when each node sees different data with different attack types.Three of the clients are simulated using samples from the Mafate dataset. They each receive only measurements that can physically be seen given the point in question — something a "real" implementation would have to record.

SolarNode observes seven channels from the solar plant instrumentation: : pv1_kW, pv2_kW, inverter_kW, pv1_voltage_V, pv2_voltage_V, battery_voltage_V, and battery_current_A.. This node sees generation and storage state directly.

DemandNodeA observes seven channels from the first residential demand point: load_total_kW, load_tv_W, load_washing_machine_W, load_fridge_W, pv1_kW, inverter_kW, and battery_voltage_V. This node sees household consumption and partial generation visibility

DemandNodeB observes seven channels from the second residential demand point: load_total_kW, load_fridge_W, battery_voltage_V, battery_current_A, pv2_kW, pv2_voltage_V, and inverter_kW. This node has stronger battery observability than DemandNodeA and a different load profile.

These three clients are non-IID by design. Each observes a different subset of physical quantities that programme distinct marginal distributions. This is however not an

artificially imposed challenge for the purpose of the test — it is merely what real world distributed smart grid nodes would actually be like

2.2 Feature Engineering

Clients are built with 7 raw measurement channels. 7 rolling residual features are appended to that — essentially the delta between the current value for each channel and its 15-step rolling mean the rolling residual is calculated causally, with past and current values only going into the feature vector so that no future information leaks. Why rolling residuals: Raw measurements often appear normal under attack, divisor data has a delay correlation. An attack shifts a channel with a constant offset which will change the raw reading but retain local trend of it — the residual gets it. A ramp attack slowly increasing upward manifests as a smoothly rising residual even if the true value remains in a reasonable range. During normal operations, the mean of a distribution of residuals will be zero; a scaling attack stretches this distribution away from zero Number of Dimensions The final feature vector per client has 14 dimensions — raw channels plus residuals. Standardization of all features using StandardScaler fitted only to the training data. The scaler never has the chance to get test data; therefore, no leakage happens from test into training.

2.3 Attack Injection Per Client

An attack is injected in a unique way on each node. In practice, real distributed attacks do not have the same impact on each node; indeed, a different threat model applies to each of them based on what data they hold and what an attacker stands to gain by corrupting it.
SolarNode receives two attack types. Scaling Attacks: These attacks scale selected channels by random factors α U[1.4, 3.0], which can be interpreted as manipulating sensor gains on generation equipment. Ramp attacks add a linearly growing perturbation β·t where β U[0.002, 0.02] — simulating continuous drift in the sensor measurement due to gradual downgrade of device quality or malicious manipulation by attackers.
DemandNodeA receives bias injection attacks. We add to the load measurement channels a constant offset δ ~ N(0, σ_feat)·U[2.5, 5.0] — simulating a scale whose sensor is being tampered so that it systematically reports an altered value. Each channel then has an applied perturbation… the magnitude of which is linked to each channel's normalised std, so that the absolute shape isn't directly possible (large enough to be relevant but not totally absurd).

DemandNodeB A replays attack with bias injections Replay replaces present channel reads with old readings at a random distance shifted by lag τ ~ U[300, 3000] steps, mimicking an adversary who replays historical operational data and conceals a current anomaly. This injection adds a constant offset over the re-played values, complicating the separation of normal operation and attack.

All three clients inject at a 35% attack ratio. Attack samples are labelled 1, normal samples 0.

2.4 . Federated Averaging Protocol

Protocol: We operate for R = 10 communication rounds with E = 5 local training epochs. You are all modelling the same Multi-Layer Perceptron architecture: input (14 features) → hidden layer 1(128 units, ReLU) → hidden layer 2(64 units, ReLU) → output (1 unit, sigmoid). Loss_function = binary_crossentropy - Adam optimizer with conquer learning rate=0.001 and batch size=200

Each round proceeds in four steps. First, every client initialises its local MLP with the current global weights and trains for 5 epochs on its own labelled data. Second, each client sends its updated weight matrices and bias vectors to the aggregation server — no raw measurements are transmitted at any point. Third, the server computes the weighted average of all client weights:

$$\mathbf{W}_l^{global} = \sum_{k=1}^{K} \frac{n_k}{N} \mathbf{W}_l^{(k)}, \quad \mathbf{b}_l^{global} = \sum_{k=1}^{K} \frac{n_k}{N} \mathbf{b}_l^{(k)} \tag{1}$$

where n_k is the number of training samples at client k and N is the total across all clients. Fourth, the aggregated global weights are broadcast back to all clients and replace their local weights before round R+1 begins.

Raw client data never leaves the local node at any point in this process. The only information that moves across the network is the model weight arrays — which contain no recoverable measurement values.

2.5 Baselines and Evaluation

Two baselines frame what the federated model can and cannot achieve.

The centralized baseline trains a single MLP on all three clients' data pooled into one dataset. This is the upper bound — full data access, no privacy. It shows what perfect information sharing would give you.

The local-only baseline trains three independent MLPs, each on their own client data only, with no aggregation. This is the lower bound — full privacy, no collaboration. It shows what each node could do entirely on its own.

The federated model sits between these two. How close it gets to the centralized upper bound — and how far it rises above the local-only lower bound — quantifies both the cost of privacy and the benefit of federation.

Each client uses an 80/20 stratified train-test split. StandardScaler fitted on training data only per client. For the centralized baseline, a unified scaler is fitted on pooled training data. Metrics reported: Accuracy, F1-score (macro-averaged), and AUC-ROC per client and globally. The federated model is additionally tracked round-by-round on validation F1 to show how detection performance builds across the 10 communication rounds.

III. Mathematical Modeling

3.1. Rolling Residual Features

Each client's 14-dimensional feature vector combines 7 raw measurements with 7 rolling residuals. The residual for channel j at time t:

$$r_j^{(k)}(t) = s_j^{(k)}(t) - \hat{s}_j^{(k)}(t) \quad (2)$$

This should remain close to zero level when everything is fine. Each of these attacks claim a bias attack pushes it by the offset injected. It notches up courtesy of a ramp attack. A scaling attack pulls the distribution away from zero. By itself the raw channel value looks perfectly normal — but the residual does not.

3.2 Cyber Attack Models

Four physically distinct injection functions target different clients.

Scaling (SolarNode—PV power channels):

$$\tilde{s}_j(t) = \alpha \cdot s_j(t), \quad \alpha \sim \mathcal{U}[1.4,\ 3.0] \quad (3)$$

Simulates sensor gain manipulation. The temporal shape of the signal is preserved — only the magnitude changes.

Ramp (SolarNode — PV voltage channel):

$$\tilde{s}_j(t_i) = s_j(t_i) + \rho_i, \quad \rho_i = \frac{2\sigma_{pv1V}}{n_{atk}} \cdot i \quad (4)$$

Linear from 0 to 2σ over the course of the attack window. Runs a slowly degrading or unsafe voltage sensor simulation This perturbation is invisible at just a single timestep but accumulates over time to form a detectable trend in the rolling residual.

Bias injection (DemandNodeA — load channels):

$$\tilde{s}_j(t) = s_j(t) + \delta_j, \quad \delta_j = \sigma_j \cdot \gamma \cdot \xi, \quad \gamma \sim \mathcal{U}[2.5,\ 5.0],\ \xi \in \{-1, +1\} \quad (5)$$

Constant signed offset scaled by each channel's standard deviation. Simulates a tampered meter reporting a systematically shifted reading. Both positive and negative injection directions are included.

Replay Attack (DemandNodeB â€" load channels):

$$\tilde{s}_j(t) = s_j\big((t+\tau) \bmod T\big), \quad \tau \sim \mathcal{U}_{\mathbb{Z}}[300,\ 3000] \quad (6)$$

$$\tilde{s}_{bat}(t) = s_{bat}(t) + \delta_{bat}, \quad \delta_{bat} = \sigma_{bat} \cdot \gamma_{bat}, \quad \gamma_{bat} \sim \mathcal{U}[2.0,\ 4.0] \quad (7)$$

Equations (5) replaces present-day observations with underlying real values, but using with these replaced observations genuine data from the operation, albeit of a completely incorrect time period. We define the battery channel in equation (6) by additionally adding a top bias. Combined they form the most difficult detection challenge: real values that are time displaced but also have an additional constant offset.

3.3 Federated Averaging Aggregation

Client k contributes to aggregation proportionally to its dataset size:

$$w_k = \frac{n_k}{N}, \quad N = \sum_{k=1}^{K} n_k \quad (8)$$

After each local training round the server aggregates all client weights:

$$\mathbf{W}_l^{global} = \sum_{k=1}^{K} w_k\, \mathbf{W}_l^{(k)} \quad (9)$$

$$\mathbf{b}_l^{global} = \sum_{k=1}^{K} w_k\, \mathbf{b}_l^{(k)} \quad (10)$$

Larger clients influence the global model more. SolarNode has the most samples, so its weights carry the largest share of each aggregation. The global weights are broadcast back to all clients before the next round begins — no raw data moves at any point in this process.

IV. Results and Discussions

4.1 Dataset and Partition Structure

The merged Solar-Demand files yield 89,179 temporally aligned measurement rows after application of inner join and missing-value removal. These are divided over three federated clients — SolarNode, DemandNodeA, DemandNodeB — with each of them only getting the channels it can physically see. Attack injection at 35% per client, thus A node generates 31,212 synthetic attack samples. Each client contains 96,312 training samples and 24,079 test samples after the 80/20 stratified split. The three of them are in fact different. Generation and storage channels in SolarNode; scaling & ramp attacks — multiplicative gain manipulations plus progressive voltage drift. Demand node A, has household consumption and limited generation a view of the system with bias injected into load nodes. You are trained up to the cleft point in October, 2023. DemandNodeB has more intensive battery observability and is exposed to the strictest condition — replay attacks with bias injection across both load and battery channels. This is not artificial heterogeneity. It corresponds to the real observation at smart grid distributed nodes and reflects realistic attacks on such nodes.

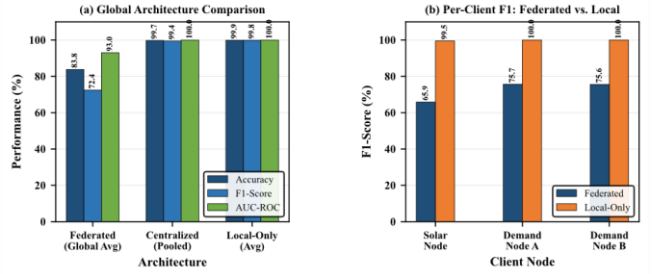

FIGURE1: Three-way architecture performance comparison: (a) global accuracy, F1-score, and AUC-ROC for federated (global average), centralized (pooled), and local-only (average) architectures; (b) per-client F1-score comparison between federated and local-only models across SolarNode, DemandNodeA, and DemandNodeB.

4.2 Three-Way Comparison — Federated vs Centralized vs Local-Only

Table I presents the final detection performance of all three architectural variants across all client nodes, evaluated on held-out test sets after 10 FedAvg communication rounds.

Table I: Three-Way Performance Comparison â€" Federated, Centralized, and Local-Only Models

Node / Architecture	Accuracy (%)	F1-Score (%)	AUC-ROC (%)
Federated â€" SolarNode	78.43	65.87	88.66
Federated â€" DemandNodeA	84.41	75.67	95.77
Federated â€" DemandNodeB	88.53	75.56	94.47
Federated â€" Global Average	**83.79**	**72.37**	**92.97**
Centralized (Pooled)	99.71	99.43	99.96
Local-Only â€" SolarNode	99.72	99.46	99.88
Local-Only â€" DemandNodeA	99.99	99.98	100.00
Local-Only â€" DemandNodeB	99.99	99.98	100.00

The centralized model hit 99.43% F1. The federated global model hit 72.37% F1. That 27.06 percentage-point gap is the cost of privacy on this dataset — the number this paper was built to measure.

Why is centralized so much better? It sees all three node types at once during training. It learns joint patterns across PV generation, battery voltage, and load channels simultaneously. When a scaling attack at SolarNode coincides with normal load patterns at DemandNodeA, the centralized model can use the cross-node relationship to sharpen its boundary. The federated model never sees that joint pattern. It aggregates locally specialised weights from nodes whose measurement distributions and attack types differ fundamentally — and weight averaging across fundamentally different local models dilutes all three.

The local-only result is the surprising one. Local-only models hit 99.46–99.98% F1. That is better than the federated model at every single node. The reason is attack specificity. Each client faces one type of attack. SolarNode only sees scaling and ramp attacks. DemandNodeA only sees bias. DemandNodeB only sees replay-bias. A model trained only on SolarNode data memorises the exact feature signatures of scaling and ramp attacks without interference from the bias and replay patterns at other nodes. When FedAvg averages those specialised weights together, the node-specific boundaries get diluted. The global model becomes a generalist that handles all attack types partially rather than handling any attack type well.

SolarNode has the worst federated performance — F1 = 65.87%, AUC = 88.66%. Scaling attacks have a multiplicative structure. The resulting feature distribution is asymmetric. When averaged with weights trained on additive bias and replay attacks at the other nodes, the scaling-specific boundary degrades more than the additive attack boundaries do.

Figure 1 shows the performance comparison visually — panel (a) gives global accuracy, F1, and AUC-ROC across all three architectures; panel (b) shows the per-client F1 gap between federated and local-only.

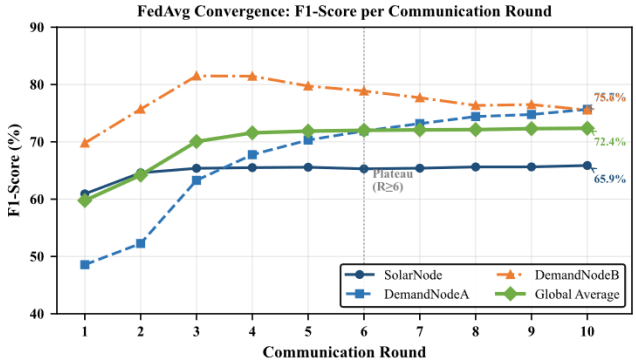

FIGURE 2 FedAvg convergence analysis — F1-score per communication round for SolarNode, DemandNodeA, DemandNodeB, and global average across 10 communication rounds. The convergence plateau beginning at Round 6 is marked by a vertical dotted line.

4.3 Round-by-Round Convergence

The F1-score per client and globally across all 10 FedAvg rounds is shown in Figure 2. Global F1 began in round 1 at 59.76% for and progressed to 72.37% on round ten — giving an overall gain of +12.61 percentage-points 59.76% → 70.05%, +10.29pp gain in the first three rounds, Rounds 4 through 10 only added +2.32pp more This is the standard FedAvg pattern for non-IID data, where early in training rapid improvement occurs as the global model captures the main patterns, then diminishing returns happen since later improvements require some node-specific fine-tuning that cannot be captured by simply averaging globals.

The three clients converged differently. DemandNodeA gained most distance — from 48.56% at round 1 to 75.67% at round 10, a gain of +27.11pp Cherry-picking some dimensions to offset in the representation of clients at a node is seemingly quite advantageous — particularly if it receives partial reinforcement during aggregation, too; for instance, because the motif signature of DemandNodeA is sufficiently similar to those of other clients that incoming global weights actually help. DemandNodeB seeing a high of 81.50% in round 3 then decreasing down to 75.56% by round 10. The global weights used in the training of DemandNodeB early

on start to clash with the DBDs as patterns from the other clients take over for larger rounds. Across all 10 rounds, SolarNode remained almost constant — 60.92% − > 65.87%. The scaling-ramp boundary creates an equilibrium very early that aggregation can never cross.Global F1 hardly shifted from round 6 with: 72.01%, 72.09%, 72.13%, 72.30% and finally up to: 72.37%. We demonstrate that practical convergence of this partition under standard FedAvg is to a distance with O(1) radius relative to the imbalance between minimizing f and maximizing g, up to tens of rounds (per-iteration equal power spectra). Adding more rounds incurs communication cost without sufficiently improving detection.

Compute

Training for ten full FedAvg rounds over all three clients takes 8–12 mins on standard CPU hardware. Centralized baseline: pooling all client data & train 50 epochs on a partition of 288,936 samples — takes ~3–5 mins. Local-only models run for each in 1–2 minutes. Even the federated averaging aggregation step itself — calculating weighted averages of weight matrices and bias vectors — takes < 1 second per round. Federation decreases the training time but does not add potential aggregation overhead. All the results are completely reproducible due to fixed seed 42.

V. CONCLUSION

Federated learning is great for multi-party training while avoiding heavier privacy losses so — how much detection power you are really trading away depends on the data, the nodes and attack conditions. This tradeoff was measured on real data collected from the isolated microgrid at Mafate, partitioned over three heterogeneous federated clients with node-specific attack types representative of true non-IID conditions. 89,179 observation signals that are divided into a solar plant node and two residential demand nodes — each with only its physically observable channels and each facing its own attack. We run FedAvg for 10 rounds with 5 local epochs at each round. The global model improved from 59.76% F1 score in round 1 to 72.37% by the end of round 10. (this advancement occurred mainly within the first three rounds. By round 6 the curve was flat.

The honest answer to the question this paper posed, what does the privacy really cost in this setting, is the 27.06 percentage-point gap between our federated model (72.37% F1) and a fully centralized baseline (99.43% F1). It is expensive in real heterogeneous microgrid data with node-specific attack types. This centralized model learns shared representations between the three types of nodes. The federated model averages the weights of nodes with completely different attack signatures — scaling attacks at SolarNode, a bias injection at DemandNodeA, and a replay-bias at DemandNodeB — where the averaging mix all three. The biggest takeaway was the local-only outcome. The models trained on data solely from their own node achieve 99.46–99.98% F1, out competing the federated model at every single node. Under strong non-IID conditions, standard FedAvg does not only fail to improve local performance. What it does is its actively corrupts it by taking decision boundaries that are supposed to be separate and mixing them. And that is the central challenge you need to solve for federated smart grid cybersecurity before it is ever deployed.

DemandNodeA converged the most, ranging from 48.56% to 75.67% across 10 rounds (27.11pp gain.) According to them, its bias injection attack strategy is a bit encouraged by other clients with similar attack approaches during the aggregation process. DemandNodeB initially had high values at round 3 then dropped, as global weights from other nodes began affecting its replay-bias boundary. For the whole time, SolarNode hardly moved — the scaling-ramp attack framework at that node is very heavily not helped by what different clients do. Three things come next. We found that by simply using FedProx or SCAFFOLD in place of standard FedAvg this gap can be far smaller than 27.06pp — with the goal being to not have to centralize data. The weight transmission step needs to be augmented with differential privacy mechanisms so that a formal mathematical guarantee of privacy can be given, instead of relying on the fact that not sharing raw data provides implicit protection. Of course, the framework should be evaluated with dynamic participation — because actual smart grid communication links are not persistent and never have partial-participation rounds been tested on realistic operational microgrid data.

REFERENCES

[1] Chandu, Gutti, Thumula Karthik, and Balbudhe Parag. "Federated learning for distributed IoT security: A privacy-preserving approach to intrusion detection." IEEE Access (2025).

[2] Fan, Yingrui, Junbo Zhao, and Meng Yue. "Adaptive Privacy Preserving Federated Learning for Virtual Power Plant Cyberattack Detection." IEEE Transactions on Smart Grid (2025).

[3] Chen, Xiaolu, Chenghao Huang, Yanru Zhang, and Hao Wang. "Privacy-Preserving Generation Fraud Detection for Distributed Photovoltaic Systems: A Solar Irradiance-Fused Federated Learning Framework." IEEE Transactions on Smart Grid (2026).

[4] Kausar, Firdous, Sambrdhi Deo, Sajid Hussain, and Zia Ul Haque. "Federated deep learning model for false data injection attack detection in cyber physical power systems." Energies 17, no. 21 (2024): 5337.

[5] Zhang, Guihai, and Biplab Sikdar. "A novel adversarial FDI attack and defense mechanism for Smart Grid demand-response mechanisms." IEEE Transactions on Industrial Cyber-Physical Systems 2 (2024): 380-390.

[6] Alrashdi, Ibrahim, and Salahaldeen Duraibi. "Intelligent cybersecurity management in industrial IoT system using attribute reduction with collaborative deep learning enabled false data injection attack detection approach." Scientific Reports (2025).

[7] Fu, Sha, Ping Li, Hong Sang, and Minxin Zhao. "Replay attack detection based on smooth watermarking and encoding and decoding." Transactions of the Institute of Measurement and Control 48, no. 3 (2026): 411-420.

[8] Hamdi, Najet. "Investigating the efficiency of a federated Learning-Based intrusion detection system for smart grid." SN Computer Science 6, no. 3 (2025): 245.

[9] Guo, Ping, Cheng Bai, Mingxing Zhang, and Puwadol Oak Dusadeerungsikul. "Clustered Federated Learning with Heterogeneous Differential Privacy on Non-IID Data." Computer Communications (2025): 108339.

[10] Hossain, Md Alamgir. "Deep Q-learning intrusion detection system (DQ-IDS): A novel reinforcement learning approach for adaptive and self-learning cybersecurity." ICT Express (2025).

[11] Olanrewaju-George, Babatunde, and Bernardi Pranggono. "Federated learning-based intrusion detection system for the internet of things using unsupervised and supervised deep learning models." Cyber Security and Applications 3 (2025): 100068.

[12] Ferrag, Mohamed Amine, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke. "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning." IEEe Access 10 (2022): 40281-40306.

[13] Toker, Onur, and Mohammad Reza Khalghani. "Cyber anomaly detection design for microgrids using an artificial intelligent-based method." In 2022 North American Power Symposium (NAPS), pp. 1-5. IEEE, 2022.

[14] Gupta, Kirti, Isha Chaudhary, Sai Sree Laya Chukkapalli, Anupam Joshi, and Bijaya Ketan Panigrahi. "A privacy-preserving anomaly diagnosis scheme for AC microgrids." Sustainable Energy, Grids and Networks 38 (2024): 101346.

[15] Deng, Ruilong, Gaoxi Xiao, Rongxing Lu, Hao Liang, and Athanasios V. Vasilakos. "False data injection on state estimation in power systems—Attacks, impacts, and defense: A survey." IEEE transactions on industrial informatics 13, no. 2 (2016): 411-423.

[16] Wang, Wu, Fouzi Harrou, Benamar Bouyeddou, Sidi-Mohammed Senouci, and Ying Sun. "Cyber-attacks detection in industrial systems using artificial intelligence-driven methods." International journal of critical infrastructure protection 38 (2022): 100542.

[17] Al-Abassi, Abdulrahman, Hadis Karimipour, Ali Dehghantanha, and Reza M. Parizi. "An ensemble deep learning-based cyber-attack detection in industrial control system." Ieee Access 8 (2020): 83965-83973.

[18] Agrawal, Shaashwat, Sagnik Sarkar, Ons Aouedi, Gokul Yenduri, Kandaraj Piamrat, Mamoun Alazab, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. "Federated learning for intrusion detection system: Concepts, challenges and future directions." Computer Communications 195 (2022): 346-361.

[19] Bouayad, Abdelhak, Hamza Alami, Meryem Janati Idrissi, and Ismail Berrada. "Lightweight federated learning for efficient network intrusion detection." IEEE Access 12 (2024): 172027-172045.

[20] Raza, Mubashar, Muhammad Jasim Saeed, Muhammad Bilal Riaz, and Muhammad Awais Sattar. "Federated learning for privacy-preserving intrusion detection in software-defined networks." Ieee Access 12 (2024): 69551-69567.

[21] Ferreira, Diogo MVP, Pedro MS Carvalho, and Marija D. Ilić. "A Kalman filter approach for state estimation in weakly monitored active distribution networks." Sustainable Energy, Grids and Networks 39 (2024): 101423.