EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Verifiable Privacy-Preserving Federated Learning for Smart Grids

Liyan Zhou[1]

[1]School of Information Engineering, Jiangsu Maritime Institute, Nanjing, 211170, China

Abstract

To address model-update leakage, malicious-client participation, and the difficulty of verifying aggregation results in smart-grid federated learning, this paper proposes a dynamic trust-driven verifiable privacy-preserving federated learning method, DT-VPFL. The method integrates training quality, update consistency, historical trust, and verification status for client selection and trust weighting, and combines random masking, threshold secret sharing, verifiable secret sharing, digital signatures, and homomorphic hashing to protect local updates and provide two-layer result verification. Experiments on the SGCC dataset show that, without attacks, DT-VPFL achieves ACC, F1-score, and AUC values of 0.9609, 0.8547, and 0.9758, respectively. When 40% of clients are malicious, the AUC remains 0.9246, compared with 0.7825 for FedAvg; with 20% malicious clients, the detection rate reaches 0.956. Under model-update inversion attacks, NMSE increases from 0.148 to 0.932, while the Pearson correlation coefficient decreases from 0.872 to 0.091. Independent synthetic-input tests reproduce the secure consistency score within 1.63×10^{-10} and bound quantization error; 100 of 100 single-coordinate tamper trials are rejected.

Keywords: smart grid; federated learning; privacy preservation; dynamic trust; verifiable secure aggregation; malicious clients

Copyright © 2026 Liyan Zhou, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetsis.14817

*Corresponding author. Email: 20158106@jmi.edu.cn

1. Introduction

1.1 Background and Significance

With the widespread deployment of smart meters, distributed energy resources, edge computing, and advanced communication technologies, conventional power grids are rapidly evolving toward digitalization, networking, and intelligence. Smart grids continuously generate large volumes of sensitive data, including user loads, device operating states, distributed energy outputs, and fault alarms, which provide essential support for intelligent applications such as load forecasting, anomaly detection, energy scheduling, and state estimation. However, conventional centralized machine learning generally requires distributed data to be collected at a central server, increasing the risks associated with data transmission and centralized storage while making it difficult to satisfy privacy and security compliance requirements for power data. Federated learning (FL) enables collaborative learning without moving raw data off local devices by performing model training locally and exchanging only model updates, thereby providing a new technical approach to cross-user and cross-region data collaboration and joint modeling [1-2].

The distributed nature of federated learning is well aligned with the data organization of smart grids, and FL has been applied to load forecasting, anomaly detection, electricity theft detection, risk assessment, and energy management [3]. Meanwhile, smart-meter data are highly privacy-sensitive, and fine-grained electricity consumption records may allow attackers to infer users' daily routines and behavioral patterns [4]. Recent studies have applied FL to distributed energy-consumption forecasting [5], false-data-injection attack detection [6], and smart-grid risk assessment [7], further demonstrating its ability to exploit multi-party data while keeping raw data local.

However, keeping raw data local does not make the federated learning process inherently secure. Smart-grid FL

remains vulnerable to model-update leakage, malicious clients, model poisoning, inference attacks, and communication tampering [8]. Existing studies have enhanced model-parameter confidentiality through multi-key homomorphic encryption [9] and combinations of homomorphic encryption with edge federated learning [10]. Nevertheless, these mechanisms mainly address whether model updates can be accessed, rather than whether submitted updates are trustworthy or whether the server performs aggregation correctly. Therefore, federated learning mechanisms that jointly provide model-update privacy, client trustworthiness, data integrity, and aggregation verifiability are of both theoretical and practical importance for improving the security, reliability, and deployability of distributed intelligence in smart grids.

1.2 Related Work

To address privacy leakage in federated learning, existing studies mainly employ differential privacy, homomorphic encryption, secure multiparty computation, and secret sharing. Xue and Yan introduced adaptive differential privacy into edge federated learning to balance privacy protection and model performance [11]. Bonawitz et al. proposed a dropout-resilient secure aggregation protocol that allows the server to obtain only the aggregate of client model updates [12]. Xu et al. further proposed TAPFed, which improves privacy protection in multi-aggregator environments through threshold secure aggregation [13]. On this basis, research has gradually expanded from parameter confidentiality to input validity and aggregation-result trustworthiness. Guo et al. combined input verification, gradient masking, and secret sharing to improve dropout resilience in secure aggregation [14], while Zhang et al. further used verifiable secret sharing and homomorphic hashing to protect model updates and verify aggregation results [15].

Privacy threats in federated learning span the data, model, and training-process levels [16]. Existing studies have shown that shared gradients may be exploited to reconstruct clients' original training data [17-18], indicating that keeping data local alone is insufficient for strong privacy protection. To improve training trustworthiness, Xu et al. proposed VerifyNet, which protects local gradients while enabling clients to verify server-side aggregation results [19]. Hahn et al. proposed VerSA, a lightweight mechanism for secure aggregation and result verification [20]. Subsequent schemes such as VFL, VOSA, and multi-key verifiable federated learning further improved security through gradient blinding, aggregation verification, and key management [21-24]. More recent studies have reduced the computational cost of verifiable secure aggregation using lightweight homomorphic encryption, homomorphic signatures, and low-overhead verification mechanisms [25-26].

In addition to privacy and verifiability, smart grids face Non-IID client data, dynamic client participation, and malicious nodes. Cheng and Liu addressed heterogeneous edge environments using dynamic clustering and adaptive training to improve collaboration among clients with different capabilities [27]. Shabbir et al. introduced anomaly-aware dynamic clustering to improve the robustness of federated models against data- and model-poisoning attacks in smart grids [28]. These studies indicate that federated learning security is evolving from standalone cryptographic protection toward coordinated privacy, robustness, and client trust management.

Recent work outside smart grids also links trust and robust aggregation. Li et al. [32] study poisoning resistance in partially trusted environments, exploiting the presence of trusted participants. Xiao [33] combines representation-based adaptive trust with hard filtering and soft weighting on image-classification benchmarks. These studies show that adaptive trust and trust weighting are established directions. DT-VPFL focuses on their integration with authenticated secure aggregation and attributable verification feedback; its experiments do not establish superiority over these additional defenses.

In summary, although substantial progress has been made in model-update privacy, secure aggregation, and result verification, three limitations remain. First, cryptographic verification mainly determines whether identities, secret shares, or aggregation results comply with the protocol, but it is difficult to identify malicious model updates that carry valid signatures. Second, secure aggregation requires individual client updates to remain hidden, whereas malicious-client detection often relies on gradient or model differences, creating an inherent tension between the two objectives. Third, trust-aware defenses already adapt client contributions, but their integration with privacy-preserving, cryptographically verifiable aggregation requires explicit treatment of feedback timing, leakage, and protocol overhead. Therefore, a dynamic-trust-driven verifiable privacy-preserving federated learning method tailored to smart grids is needed.

1.3 Research Motivation and Contributions

Although existing federated learning schemes can protect model updates through secure aggregation and verification mechanisms, they still struggle to identify clients whose identities are legitimate and whose messages pass verification, but whose updates are malicious. In addition, data heterogeneity, dynamic node participation, and poisoning attacks further complicate trustworthy aggregation in smart grids. Prior studies indicate that dynamic trust management [29] and privacy-preserving federated learning for heterogeneous smart-grid data [30] have promising potential. Motivated by these observations, this paper proposes a dynamic trust-driven verifiable privacy-preserving federated learning method, termed DT-VPFL. The main contributions are as follows:

1. A dynamic trust-driven framework for smart grids couples client admission and aggregation weights with historical, quality, consistency, and attributable verification signals in one iterative workflow.

2. The methodological contribution is the closed-loop coupling of multidimensional trust, client selection, continuous weighting, and verification feedback. Trust weighting alone is not claimed as a new primitive, and a failed server aggregate is distinguished from an attributable client failure.

3. Established random masking, threshold and Feldman verifiable secret sharing, authenticated encryption, digital signatures, and homomorphic hashing are integrated into this workflow. These cryptographic building blocks are reused rather than newly invented. Evaluation on SGCC data examines predictive performance, poisoning resistance, verification, and reported system overhead.

2. System Model and Problem Formulation

2.1 System Model

As shown in Figure 1, the considered federated learning system consists of multiple smart-grid clients and an aggregation server. Each client holds private local data and performs model training locally, while the server receives privacy-protected and trust-verified model updates, performs secure aggregation, and updates the global model.

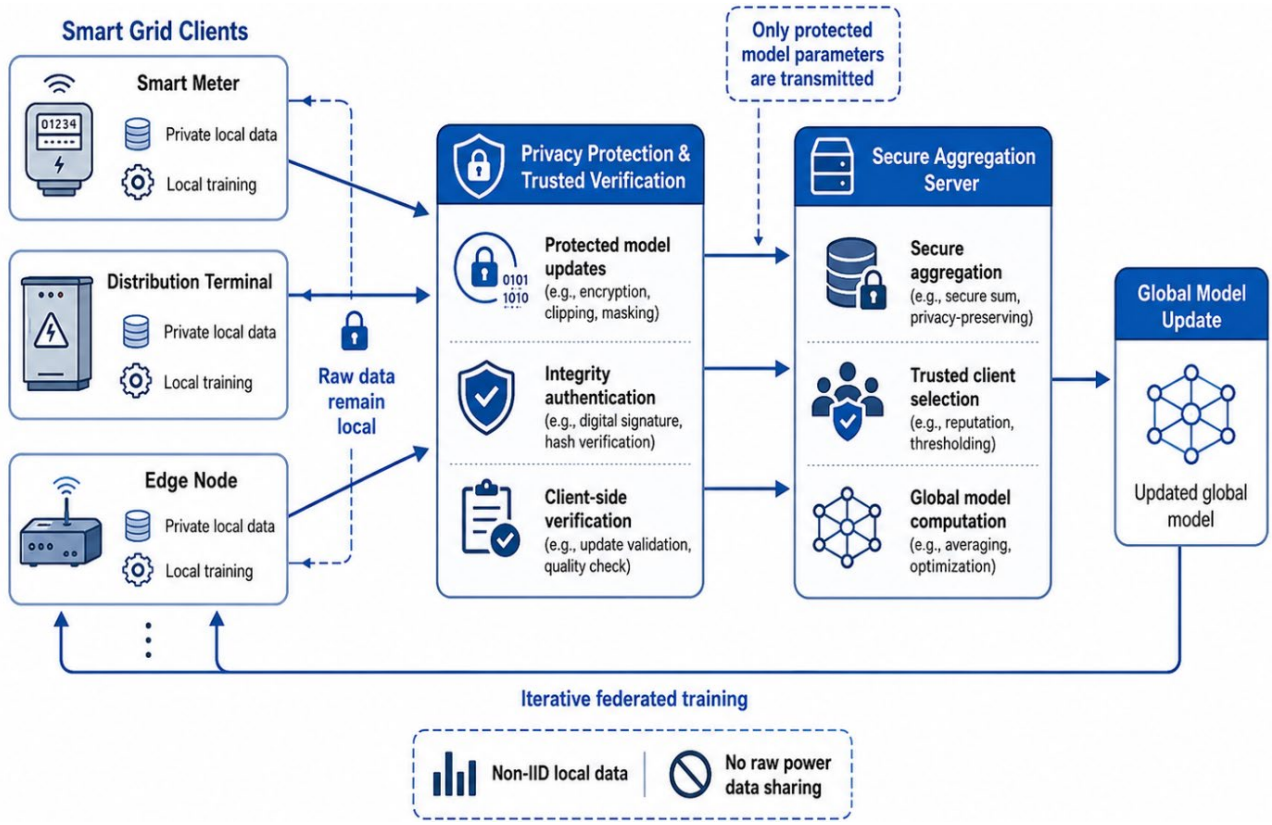

Figure 1. System model of verifiable privacy-preserving federated learning for smart grids

Consider a federated learning system consisting of N smart-grid clients and an aggregation server S. A client may represent a smart meter, distribution terminal, or edge node, and the client set is defined as:

$$\mathrm{U} = \{U_1, U_2, \ldots, U_N\} \quad (1)$$

Client U_i holds the private local dataset:

$$D_i = \left\{\left(x_{ij}, y_{ij}\right)\right\}_{j=1}^{n_i}, \quad i \in \{1, 2, \ldots, N\} \quad (2)$$

where n_i denotes the number of local samples. Different clients share the same feature space, but due to differences in regional load characteristics, user types, and device states, their data are generally non-independent and identically distributed (Non-IID). Clients train only on local data, and raw data always remain local. The local objective function is defined as:

$$F_i(w) = \left(1/n_i\right) \sum_{j=1}^{n_i} \ell\left(w; x_{ij}, y_{ij}\right) \quad (3)$$

where w denotes the model parameters and $\ell(\cdot)$ denotes the task loss function. The global optimization objective of federated learning is:

$$\min_w F(w), \quad F(w) = \sum_{i=1}^{N} (n_i / n_{tot}) F_i(w) \qquad (4)$$

In each training round, the server broadcasts the current global model to participating clients. After local training, each client generates a local model update, applies privacy protection and integrity authentication, and uploads the protected update. The server aggregates only updates that satisfy the security and trust requirements and sends the updated global model to legitimate clients until convergence.

2.2 Threat Model

The following security threats are considered in the open-network environment of a smart grid.

1) Honest-but-curious aggregation server. The server follows the federated learning protocol for training and aggregation but may attempt to infer local electricity-consumption data from uploaded model updates and intermediate information. The intended leakage comprises the accepted aggregate, explicitly released trust statistics, and public participation metadata. Individual-update confidentiality is conditional on the secure-computation and aggregation assumptions stated in Sections 3.2 and 3.6.1.

2) Malicious clients. Some terminals may be compromised and may reduce global-model performance through data poisoning, model poisoning, or abnormal parameter updates. Because a malicious client can use a legitimate identity and key to generate a valid signature, identity authentication alone cannot determine whether its model update is trustworthy.

3) Communication attackers. External attackers may eavesdrop on, forge, tamper with, or replay data transmitted between clients and the server. Therefore, the authenticity and integrity of model updates, secret shares, and aggregation results must be ensured.

4) Partial entity collusion. The aggregation server may share the information it obtains with a small number of malicious clients. Let the colluding client set be C, and suppose its size is below the secret-sharing reconstruction threshold k,

$$|\mathsf{C}| < k \qquad (5)$$

The condition in Eq. (5) protects unreconstructed shares; it must be combined with at least two honest contributors in a released aggregate and a common authenticated roster. An aggregate containing only one noncolluding contribution can reveal that contribution after subtraction. Threshold-breaking collusion and arbitrary malicious-server privacy attacks are outside the present analysis; output tampering is separately checked by result verification.

2.3 Design Objectives

The proposed method is designed to satisfy the following four objectives under the above threat model.

(1) Privacy. Raw data remain local, and individual updates are concealed subject to the stated leakage and collusion assumptions. Neither aggregate release nor trust-score release provides differential privacy or a general guarantee against inference.

(2) Verifiability. The source authenticity and content integrity of client model updates and secret shares are guaranteed, while clients can verify whether the aggregation result returned by the server has been generated correctly according to the protocol.

(3) Robustness. Clients that pass identity authentication but repeatedly submit anomalous or low-quality updates are dynamically evaluated, and their aggregation contribution is reduced or they are excluded to suppress the impact of malicious updates on the global model. The method also supports the dynamic dropout of a certain proportion of clients.

(4) Efficiency. While satisfying privacy protection, trust evaluation, and aggregation verification requirements, the method controls additional cryptographic computation, communication volume, and verification latency so that it remains applicable to resource- and communication-constrained smart-grid edge environments.

Accordingly, the core problem addressed in this paper is how to evaluate client trustworthiness, suppress malicious updates, and verify aggregation results in a Non-IID and dynamically participating smart-grid environment while hiding individual client model updates and maintaining acceptable computational and communication overhead. Section 3 presents the proposed dynamic trust-driven verifiable privacy-preserving federated learning method.

3. Dynamic Trust-Driven Verifiable Privacy-Preserving Federated Learning

To address local-update leakage, malicious-client participation, and insufficient aggregation-result trustworthiness in smart-grid federated learning, this paper proposes Dynamic Trust-driven Verifiable Privacy-Preserving Federated Learning (DT-VPFL). DT-VPFL integrates dynamic client trust evaluation, random masking, verifiable secret sharing, and trust-weighted aggregation into the federated training process and continuously updates client trust states using input-verification outcomes, thereby forming a closed loop of trust evaluation, privacy protection, secure aggregation, result verification, and trust feedback.

3.1 Overall Framework of DT-VPFL

As shown in Figure 2, DT-VPFL sequentially performs local training, dynamic trust evaluation, trusted-client selection, privacy-preserving update processing, verifiable secret sharing, trust-weighted secure aggregation, and aggregation-result verification. After successful aggregate verification, the server updates the global model and carries attributable verification records into the next round, thereby forming a closed-loop trustworthy training process of trust evaluation, privacy protection, secure aggregation, result verification, and trust feedback.

In training round t, the server broadcasts the global model w^t to online clients, and client U_i performs local training and obtains a local model update:

$$\Delta w_i^t = w_i^t - w^t \tag{6}$$

DT-VPFL then performs dynamic trust evaluation, trusted-client selection, model-update masking, verifiable secret sharing, secure aggregation, and result verification. Unlike conventional FedAvg, which directly aggregates all online clients, DT-VPFL allows only clients that meet the trust threshold to participate in secure aggregation and adjusts their model contributions according to their dynamic trust states. The verification results are further fed back into the next-round trust evaluation to continuously constrain abnormal participation behavior.

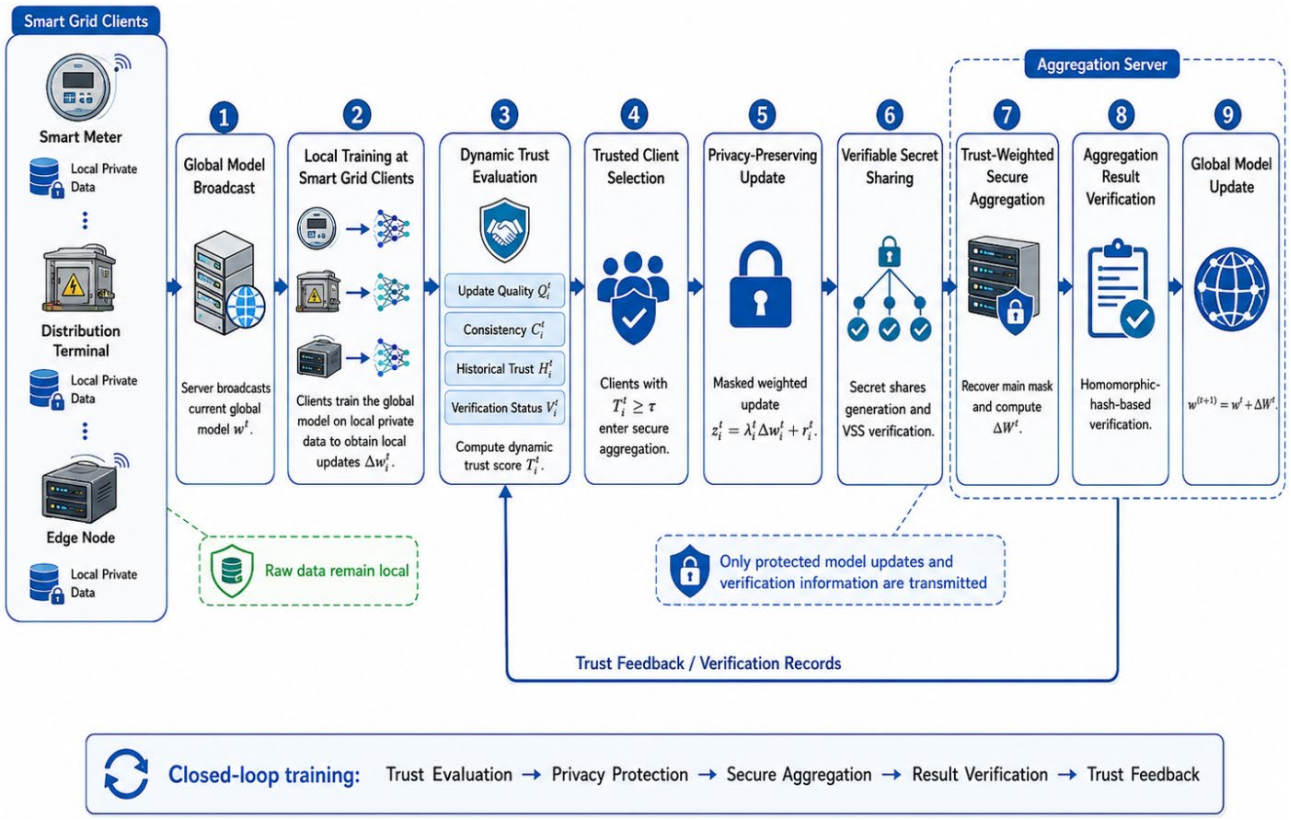

Figure 2. Overall framework of DT-VPFL

3.2 Dynamic Trust Evaluation and Client Selection

The data quality, online status, and security behavior of smart-grid terminals may change over time. Therefore, this paper adopts a multidimensional dynamic trust mechanism rather than a static honest/malicious binary assumption. Client U_i has an instantaneous trust score in round t defined as:

$$\hat{T}_i^t = \alpha Q_i^t + \beta C_i^t + \gamma H_i^t + \delta V_i^t, \quad \alpha + \beta + \gamma + \delta = 1 \tag{7}$$

where Q_i^t, C_i^t, H_i^t, and V_i^t denote local training quality, update-direction consistency, historical trust, and security verification status, respectively. First, update quality is defined according to the improvement in local loss before and after training.

The coefficients α, β, γ, and δ are nonnegative and sum to one. Scores are normalized to [0,1]. A local loss-reduction score is a heuristic quality signal; without trusted execution or verifiable training, a malicious client can misreport it. It should therefore not be interpreted as cryptographic evidence of honest training.

$$Q_i^t = \mathrm{clip}\left(\frac{L_{i,\,\mathrm{before}}^t - L_{i,\,\mathrm{after}}^t}{L_{i,\,\mathrm{before}}^t + \varepsilon}, 0, 1 \right) \tag{8}$$

where ε is a small positive constant. This definition assigns a higher quality score to an update that achieves a larger reduction in local loss, whereas the score approaches 0 when the loss deteriorates after training. Second, the previous-round global update direction $\Delta \mathbf{w}_g^{t-1}$ is used as a public

reference direction, and update consistency is computed through a secure inner product:

$$C_i^t = \frac{1}{2}\cdot\left(1+\frac{\left\langle \Delta w_i^t, \Delta w_g^{t-1}\right\rangle}{\left\|\Delta w_i^t\right\|_2 \left\|\Delta w_g^{t-1}\right\|_2 + \varepsilon}\right) \tag{9}$$

The secure scalar functionality takes the private update $\Delta \mathbf{w}_i^t$ and the authenticated public reference as inputs and releases only C_i^t. A suitable realization requires secure two-party computation of the dot product, squared norm, and fixed-point normalization; additive secure aggregation alone does not implement this functionality. The reference implementation uses two-party MASCOT in MP-SPDZ 0.4.3 [35,36], an actively secure arithmetic protocol based on oblivious transfer and authenticated secret sharing. This instantiates the active-secure computation requirement discussed in [34]. It must bind the scoring input to the subsequently committed update; otherwise a malicious client can score one vector and submit another. Section 4.8 benchmarks the standalone scoring circuit with synthetic inputs. A released score is intentional leakage and does not establish zero information leakage. Historical trust is defined as:

$$H_i^t = \frac{N_{i,\,\text{pass}}^t + \zeta}{N_{i,\,\text{part}}^t + 2\zeta} \tag{10}$$

where $N_{i,\text{pass}}^t$ and $N_{i,\text{part}}^t$ count successful attributable verifications and completed participations available before round t, and ζ is a positive pseudocount. Thus, the history term does not use the current round's future outcome. A client with no history has a history score of 0.5. The binary verification state is:

$$\mathbf{V_i^t} \in \{\mathbf{0},\mathbf{1}\} \tag{11}$$

Here, $V_i^t = 1$ means that the client has passed the checks available at admission; $V_i^t = 0$ denotes an attributable authentication, signature, or share-verification failure. A later failure removes the client from the accepted roster and is recorded before the next round. A server-side aggregate failure causes rejection of the round, but does not by itself identify a guilty client. The long-term trust score is updated by exponential smoothing:

$$T_i^t = \rho T_i^{t-1} + (1-\rho)\hat{T}_i^t, \quad 0 < \rho < 1 \tag{12}$$

Let U_t be the authenticated online candidate set. The provisional trusted set is:

$$S_t = \left\{U_i \in U_t : T_i^t \geq \tau, V_i^t = 1\right\} \tag{13}$$

where τ is the admission threshold. Invalid submissions are rejected independently of smoothing. If a selected client fails a later check, the accepted roster and weights must be agreed again before aggregation; changed weights require fresh weighted uploads. With a nonempty accepted set, local sample size and trust determine the normalized weight:

$$\lambda_i^t = \frac{n_i \left(T_i^t\right)^\eta}{\sum_{U_j \in \mathbf{S}_t} n_j \left(T_j^t\right)^\eta}, \quad \sum_{U_i \in \mathbf{S}_t} \lambda_i^t = 1 \tag{14}$$

where $\eta \geq 1$ is the trust sensitivity coefficient. A larger η further suppresses the impact of low-trust clients on the global model.

For initialization and intermittent participation, a deployment must publish its trust prior, the pseudocount ξ, and the zero-reference rule before training. Equation (9) yields a neutral consistency score of 0.5 when the reference vector is zero. If no client passes admission or the accepted set cannot meet the reconstruction threshold, the round is skipped rather than normalizing an empty set. Unexplained network dropout is recorded separately from an attributable cryptographic failure. The experiment description does not document all initialization choices, which limits exact reproduction.

Secure scoring implementation. The client supplies 18-bit signed two's-complement integers with 16 fractional bits, restricting each decoded input to [−2,2). The public reference is authenticated and uses the same scale. The circuit checks each privately supplied bit by opening b(b − 1), rejects any nonzero result, and computes the integer inner product and squared norm before fixed-point normalization. It uses 32 fractional bits in 64-bit fixed-point values and $\varepsilon = 2^{-16}$ in Eq. (9). A zero update returns 0.5; the output is clamped to [0,1]. Only the score and the prescribed validity outcome are opened. The 256-bit MPC field is distinct from the aggregation field below; statistical security is 40 bits.

The benchmark has one client and one server communicating over TLS. Fresh OT/MAC setup and preprocessing are included in each measured execution; certificate creation and circuit compilation are excluded. The public reference is fixed in the circuit for reproducibility. The bit checks enforce the scoring domain, but do not establish that a later weighted upload encodes this same input. An integrated circuit or a sound cross-protocol commitment proof is still required for that property. The supplied prototype is research code, without a production security audit.

3.3 Verifiable Privacy-Preserving Local Updates

For a trusted client, the weighted update is encoded coordinatewise as $a_i^t = Q_s\left(\lambda_i^t \Delta w_i^t\right)$ using the public scale s

defined in Section 3.4. A uniformly random vector mask r_i^t is drawn in the same finite field. The masked upload is:

$$z_i^t = a_i^t + r_i^t \,(\mathrm{mod}\, p) \tag{15}$$

The masked vector alone does not reveal the encoded local update. Clients use (k,m) threshold secret sharing, where $m = |S_t|$, over a prime field F_p. For each coordinate, the client constructs a polynomial of degree k − 1:

$$P_i^t(x) = r_i^t + \sum_{l=1}^{k-1} a_{il}^t x^l \tag{16}$$

The vector polynomial in Eq. (16) is interpreted coordinatewise. Its constant coefficient is $a_{i0}^t = r_i^t$; these polynomial coefficients are distinct from the encoded update vector a_i^t. Distinct, nonzero public evaluation points x_j identify share recipients. Feldman commitments are formed for every coefficient, including the constant term, as $C_{i\ell}^t = g^{a_{i\ell}^t}$ in a group of prime order p. Each recipient verifies, coordinatewise:

$$g^{r_{ij}^t} \overset{?}{=} \prod_{\ell=0}^{k-1} \left(C_{i\ell}^t\right)^{x_j^\ell} \tag{17}$$

A failed share check is rejected and attributed to its sender. Each signed upload binds the round, client identity, accepted roster digest, aggregation weight, quantization parameters, masked vector, and homomorphic hash. Authenticated encryption protects share exchange. All honest participants must agree on one roster and transcript before returning aggregate shares; signatures alone do not establish this agreement. Only summed shares are released to the server, never the individual shares of an honest client.

Threshold configuration under churn. Let c be the maximum colluding shareholders and D the maximum unavailable or invalid aggregate-share responses. A feasible threshold satisfies $c < k \le m - D$. The default $k = \lceil 0.6m \rceil$ tolerates at most m-[0.6m] unavailable responses while keeping c below k. For m = 10, k = 6 permits four unavailable responses and requires $c \le 5$; the accepted aggregate must still contain at least two honest contributors. A deployment should choose k from its collusion budget and a high-percentile dropout estimate, not only mean availability. If the interval is empty, the cohort must be enlarged or the round postponed. The threshold is fixed after share distribution. Late joiners enter a later round, and roster or weight changes require a fresh authenticated session; k must not be lowered during reconstruction.

3.4 Dynamic-Trust-Weighted Secure Aggregation and Result Verification

After share verification and agreement on the accepted roster, client U_j sums the valid shares for exactly that roster:

$$R_j^t = \sum_{U_i \in S_t} r_{ij}^t \tag{18}$$

Because secret sharing is additively homomorphic, once the server receives at least k valid aggregated shares, it can recover the master mask through Lagrange interpolation:

$$R^t = \sum_{U_i \in S_t} r_i^t \tag{19}$$

without separately reconstructing an individual mask r_i^t. This use of additive secret sharing follows established secure aggregation [12,15]. The server subtracts the reconstructed aggregate mask in the finite field, obtains the encoded sum, and decodes the signed representative:

$$\begin{aligned} A^t &= \sum_{U_i \in S_t} z_i^t - R^t \,(\mathrm{mod}\, p) \\ A^t &= \sum_{U_i \in S_t} a_i^t \,(\mathrm{mod}\, p) \\ \Delta W^t &= \mathrm{Decode}\left(A^t\right)/s \end{aligned} \tag{20}$$

The global model for the next round is then obtained as:

$$w^{t+1} = w^t + \Delta W^t \tag{21}$$

The next model uses the decoded aggregate. Verification is performed on the same integer sum that was securely aggregated, rather than on a separately rounded floating-point sum. Each client hashes its encoded weighted update:

$$\mathbf{h_i^t} = \mathbf{H}\left(\mathbf{a_i^t}\right) = \prod_{\mathbf{r=1}}^{\mathbf{d}} \mathbf{g_r^{a_{i,r}^t}} \tag{22}$$

Here, $Q_s(x) = \mathrm{round}(sx)$ uses a public deterministic tie rule, and negative integers are represented modulo p. The group has prime order p; the public generators g_r must be generated so that their discrete-log relations are unknown. Hash binding is computational, not information theoretic. If each weighted coordinate is bounded in magnitude by B, choosing $p > 2\,m(s\,B+1/2)$ prevents wraparound of the signed aggregate. The following equality is checked in the group:

$$\mathbf{H}\left(\mathbf{A}^t\right) \overset{?}{=} \prod_{U_i \in S_t} \mathbf{h}_i^t \tag{23}$$

The server returns the encoded aggregate At and its decoded update. Clients verify Eq. (23), the signed roster and hashes, and consistency of the next model with Eq. (21). Rounding is applied once to each weighted client update; in general, rounding a sum is not the same as summing rounded inputs. For example, with s=10, two values of 0.04 encode to 0 + 0, whereas their sum encodes to 1. The coordinatewise deviation from the unquantized weighted sum is bounded by

m/(2s) when clipping is absent and wraparound is excluded. This approximation affects training precision but does not cause a mismatch in an exact integer-domain verification. The supplementary implementation below makes these choices explicit and tests their arithmetic effects in Section 4.8. These settings describe the new reference prototype; they cannot be retrospectively attributed to the original SGCC training results.

Quantization and group configuration. The default scale is s = 65,536, with nearest-even rounding applied to each binary64 weighted coordinate. The configured bound is B = 1; nonfinite or out-of-range inputs are rejected, not silently clipped. Signed integers are encoded in the Ristretto255 scalar field [37], whose order is $p = 2^{252} + c$, where c = 27742317777372353535851937790883648493. Canonical field elements occupy 32 little-endian bytes, and residues above p/2 decode as negative. The implementation checks $p > 2m(sB + 1/2)$ before aggregation. The same encoded vector is masked and hashed, and the exact sum is verified before division by s.

The group implementation uses libsodium Ristretto255 in additive notation: $\boldsymbol{H}(\boldsymbol{a}) = \sum_{j=1}^{d} [\boldsymbol{a}_j]\boldsymbol{G}_j$, equivalent to the multiplicative homomorphism in Eq. (23). Independent bases are obtained by domain-separated SHA-512 followed by the Ristretto from-hash map; no known generator-multiple relation is introduced. Masks and polynomial coefficients use a cryptographic random source. The prototype implements coordinatewise Shamir/Feldman sharing, Ed25519 signatures, and X25519 with HKDF-SHA256 and AES-256-GCM for peer shares. Signed records and encryption contexts bind session, sender, recipient, and a canonical roster digest. Each recipient checks signed client hashes against the returned encoded aggregate. Enrollment keys are assumed pinned, and one agreed roster is assumed; a Byzantine roster-agreement service is not implemented.

The two verification layers therefore check authenticated inputs and encoded aggregation consistency under the stated assumptions. Client-attributable failures update trust; unexplained aggregate failure rejects the round and triggers diagnosis. Homomorphic hashing does not prove that a client's masked vector, scoring input, and hash encode the same private update without an additional input-consistency mechanism. The current study does not provide an end-to-end malicious-security proof of that composition.

3.5 DT-VPFL Algorithm Flow

To illustrate the execution of DT-VPFL in one federated training round, client selection, dynamic trust updating, privacy protection, verifiable secret sharing, trust-weighted secure aggregation, and result verification are summarized in Figure 3. As shown in Figure 3, the verification results are fed back to client trust states at the end of each round and further affect client selection and aggregation weights in the next round, forming a continuously iterative trustworthy training loop.

Figure 3 is a high-level schematic. The revised admission gate, finite-field encoding, authenticated roster, and failure-attribution rules are specified in Eqs. (11), (13), and (15)–(23) and the accompanying text. A post-selection dropout or verification failure requires the roster and weights to be reconciled before the aggregate is released.

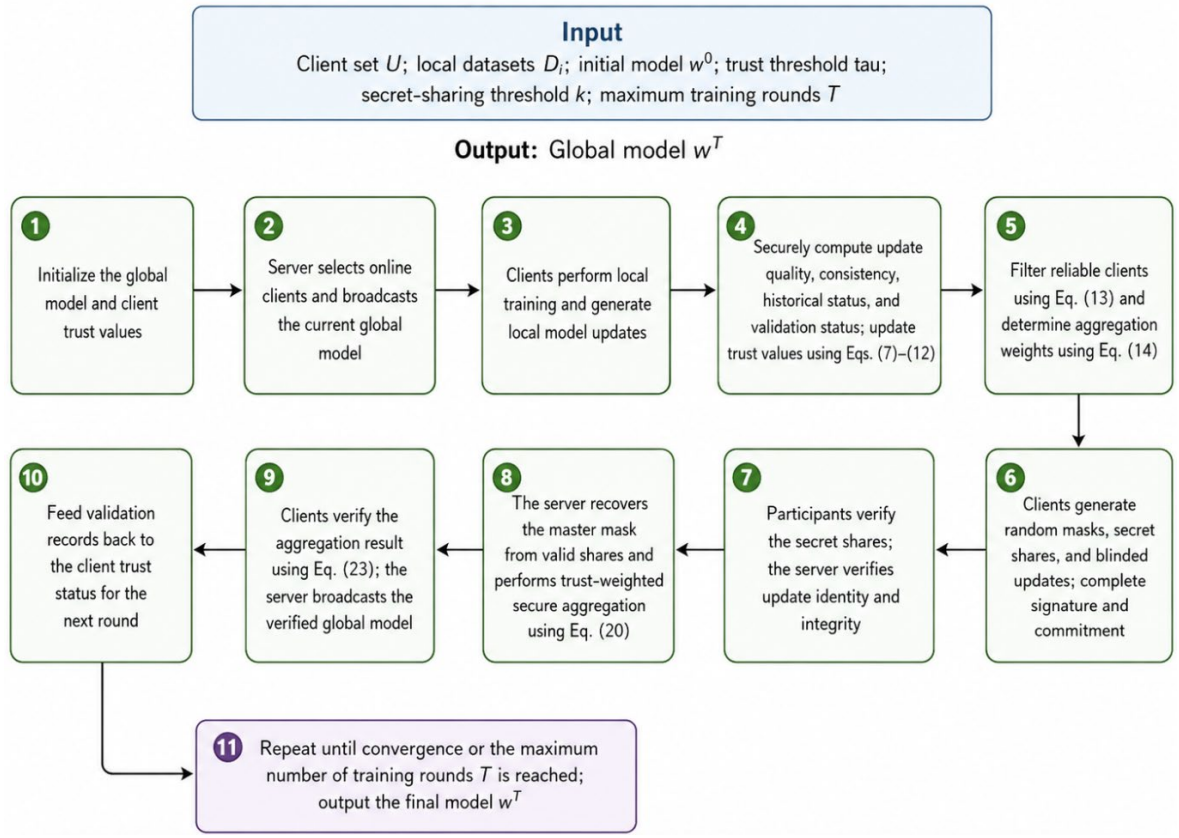

Figure 3. DT-VPFL algorithm schematic

3.6. Security and Complexity Analysis

3.6.1 Privacy

Raw data remain local. Random masking conceals individual encoded updates only while the corresponding masks remain hidden. Privacy additionally requires fewer than k colluding shareholders, at least two honest contributors to each released aggregate, fresh masks, and one authenticated roster without adversarial subset queries. Feldman commitments and homomorphic hashing add discrete-log-based computational assumptions. The intended leakage includes aggregate updates, trust scores, and participation records; repeated releases may support inference. Section 3.2 specifies and Section 4.8 tests the standalone secure scoring protocol. Its composition with aggregation still requires input binding. Public deterministic client hashes are not hiding commitments: low-entropy candidate updates can be checked by a dictionary attack. Accordingly, confidentiality claims are conditional on this leakage model and do not amount to semantic security of every individual update. The inversion experiment in Section 4.4 is evidence for the evaluated attack, rather than a universal privacy guarantee.

3.6.2 Integrity and Verifiability

Equation (17) checks consistency of shares with coefficient commitments. Authenticated encryption and signatures protect message authenticity and integrity. Equation (23) checks the encoded aggregate against authenticated client hashes under the hash-binding assumption. These checks do not establish training correctness or semantic update quality, and a signed malicious update may still pass them. Their composition also requires input binding and roster agreement; these are explicit implementation requirements rather than properties implied by signatures alone.

3.6.3 Robustness Against Malicious Clients

A legitimate identity and valid signature cannot prevent a client from deliberately generating a poisoned update. DT-VPFL therefore continuously adjusts client trust according to training quality, consistency, historical behavior, and cryptographic verification outcomes, and applies both the admission threshold in Eq. (13) and the trust weight in Eq. (14) to suppress malicious contributions. Existing studies have shown that poisoning, adversarial examples, and free riding can undermine the integrity and availability of federated training. The dynamic trust mechanism complements conventional cryptographic verification, which cannot determine the semantic trustworthiness of an update.

3.6.4 Computational and Communication Complexity

Let m be the accepted cohort size, d the model dimension, and k the reconstruction threshold. Table 1 distinguishes message counts from payload volume for the direct, coordinatewise

vector-sharing construction of Section 3.3. A vector share contains d field elements; it is not a constant-size scalar. Bit costs further multiply field-element counts by the field bit length. Trust scoring adds the separate secure-computation cost discussed below.

Table 1. Core communication cost with direct vector secret sharing

Entity	Main computational operations	**Communication** in field elements
Client	Local training, trust-metric calculation, random mask generation, secret sharing, VSS verification, and signature/verification	O(d(m + k)) per client
Aggregation server	Signature checks, aggregate-mask reconstruction, trust-weighted aggregation, and result verification	O(dm + dk) for masked uploads and k aggregate shares
Among clients	Exchange of secret shares and related verification information	O(dm²) payload in O(m²) peer-share messages

There are m(m−1) directed peer-share messages per cohort. Their payload is O(dm²) field elements, while each client publishes O(dk) coefficient commitments; naive rebroadcast of commitments can add further traffic. For n online candidates and fixed-point precision b, let C(d,b) denote the communication of one chosen secure-scalar protocol. Trust scoring adds nC(d,b), including protocol setup or preprocessing where applicable. The scalar output size does not imply constant communication: d input coordinates must be processed, with nonlinear norm and normalization operations. Section 4.8 reports separate measured scoring and aggregation costs for a specified reference implementation. Those measurements use synthetic inputs and do not validate the completeness of the original training-time accounting.

4. Experiments and Results Analysis

To evaluate DT-VPFL in smart-grid environments, experiments are conducted in terms of model performance, Non-IID adaptability, robustness against malicious clients, dynamic trust discrimination, privacy preservation and verifiability, and system overhead. Ablation experiments are also performed to assess the contribution of each core component.

4.1. Experimental Setup

4.1.1 Dataset and Data Partitioning

The State Grid Corporation of China (SGCC) public electricity-theft dataset [31] is used in the experiments. It contains daily electricity-consumption records of 42,372 users over 1,035 days, including 38,757 normal users and 3,615 abnormal users; abnormal users account for approximately 8.53% of the dataset. The pronounced class imbalance makes this dataset suitable for evaluating federated learning in smart-grid abnormal-consumption detection.

Missing electricity-consumption records are first imputed using temporal-neighborhood interpolation, and extreme values are bounded. User load data are then normalized to [0,1]. The data are split by user into training, validation, and test sets at ratios of 70%, 10%, and 20%, respectively, with no user overlap among the subsets. To simulate data heterogeneity across users in different regions, a Dirichlet distribution is used to partition the training samples among clients:

$$p_i \sim \mathrm{Dir}(\alpha_D) \tag{24}$$

where $\boldsymbol{\alpha_D}$ controls the degree of client-data Non-IID heterogeneity. Experiments use $\boldsymbol{\alpha_D}$=0.1, 0.5, and 1.0, where a smaller $\boldsymbol{\alpha_D}$ indicates stronger data heterogeneity.

4.1.2 Experimental Parameters

To prevent the complexity of the base classifier from obscuring the contribution of the federated security mechanism itself, all methods use the same lightweight one-dimensional convolutional neural network as the base classifier. Weighted binary cross-entropy is used to alleviate class imbalance between normal and abnormal samples. The SGCC experiments in Sections 4.2–4.6 use the same data partition and model initialization and are reported as five independent repetitions, with means and standard deviations. The new protocol benchmarks in Section 4.8 use their separately stated settings. The main experimental parameters are listed in Table 2.

Table 2. Experimental parameter settings

Parameter	Setting
Total number of clients (N)	20
Client participation ratio per round	50%
Global training rounds	100
Local epochs (E)	3
Batch size	64
Initial learning rate	1×10^{-3}
Optimizer	Adam
Trust threshold (τ)	0.60
Trust smoothing coefficient (ρ)	0.80
Trust sensitivity coefficient (η)	2
(α,β,γ,δ)	(0.30,0.25,0.20,0.25)
Secret-sharing threshold (k)	⌈0.6m⌉

Here, m is the accepted cohort size. The fixed defaults in Table 2 are interpretable design choices, not reported optima from a parameter search. The nonnegative weights sum to one: training quality receives 0.30, consistency and verification each receive 0.25, and historical evidence receives 0.20. This gives current evidence greater influence than history while retaining temporal information. The threshold 0.60 controls admission, $\rho = 0.80$ smooths transient fluctuations, and $\eta = 2$ increases weight separation. These choices must be calibrated for the local data distribution and attack risk; Section 4.7 analyzes their mathematical sensitivity and provides a prospective tuning procedure.

4.1.3 Baseline Methods and Evaluation Metrics

Local, FedAvg [1], SecAgg-FedAvg [12], VerifyNet [19], and VPFL [24] are selected as baseline methods. Local denotes independent client-side training; FedAvg is the standard federated averaging method; SecAgg-FedAvg adds secure aggregation to FedAvg; and VerifyNet and VPFL are representative verifiable privacy-preserving federated learning methods.

Considering the class imbalance of the SGCC dataset, Accuracy (ACC), Precision, Recall, F1-score, and AUC are used to evaluate classification performance. Malicious-client Detection Rate (DR), False Exclusion Rate (FER) for normal clients, Verification Detection Rate (VDR), training time, and communication overhead are also used to evaluate security and operational efficiency.

4.2 Model Performance and Non-IID Adaptability

The basic classification performance of different methods is first compared with no malicious clients under $\boldsymbol{\alpha_D}$=0.5, and the results are shown in Table 3.

Table 3. Model performance of different federated learning methods

Method	ACC	Precision	Recall	F1-score	AUC
Local	0.9241±0.0038	0.7462±0.0081	0.7714±0.0095	0.7586±0.0074	0.9318±0.0047
FedAvg	0.9568±0.0023	0.8461±0.0062	0.8195±0.0070	0.8326±0.0055	0.9694±0.0026
SecAgg-Fed Avg	0.9565±0.0024	0.8448±0.0065	0.8189±0.0073	0.8316±0.0059	0.9691±0.0027
VerifyNet	0.9562±0.0025	0.8437±0.0068	0.8176±0.0071	0.8304±0.0058	0.9687±0.0028
VPFL	0.9566±0.0022	0.8451±0.0060	0.8203±0.0068	0.8325±0.0054	0.9693±0.0025
DT-VPFL	0.9609±0.0019	0.8628±0.0053	0.8467±0.0059	0.8547±0.0046	0.9758±0.0021

As shown in Table 3, all federated learning methods outperform Local in overall performance. DT-VPFL achieves the highest ACC, F1-score, and AUC, reaching 0.9609, 0.8547, and 0.9758, respectively. Compared with FedAvg, DT-VPFL improves the F1-score and AUC by 0.0221 and 0.0064, respectively. SecAgg-FedAvg, VerifyNet, and VPFL perform similarly to FedAvg, indicating that secure aggregation and cryptographic verification do not significantly degrade model learning. The additional improvement of DT-VPFL mainly stems from dynamic trusted-client selection and trust weighting, which suppress low-quality local updates.

Table 4. AUC comparison under different levels of data heterogeneity

α_D	FedAvg	SecAgg-FedAvg	VerifyNet	VPFL	DT-VPFL
1.0	0.9752	0.9749	0.9745	0.9750	0.9791
0.5	0.9694	0.9691	0.9687	0.9693	0.9758
0.1	0.9436	0.9429	0.9424	0.9431	0.9617

To further analyze the impact of data heterogeneity, experiments are conducted under different Dirichlet parameters, as shown in Table 4. As α_D decreases, the performance of all methods declines. When $\alpha_D = 0.1$, the AUC of FedAvg falls to 0.9436, whereas DT-VPFL maintains an AUC of 0.9617, which is 0.0181 higher than that of FedAvg. This indicates that dynamic trust can reduce the influence of persistently low-quality local updates under strongly Non-IID conditions and improve adaptability to heterogeneous client data.

4.3. Robustness Against Malicious Clients and Dynamic Trust Analysis

4.3.1 Malicious Client Attacks

To evaluate the robustness of DT-VPFL against malicious participants, the malicious-client proportion is set to $r_m \in \{0\%, 10\%, 20\%, 30\%, 40\%\}$, and three representative attacks are implemented: Label-flipping, Sign-flipping, and Random/Gaussian Update. The AUC results of different methods under different malicious-client proportions are shown in Table 5.

Table 5. AUC under different proportions of malicious clients

Malicious clients (%)	FedAvg	SecAgg-FedAvg	VerifyNet	VPFL	DT-VPFL
0	0.9694	0.9691	0.9687	0.9693	0.9758
10	0.9321	0.9314	0.9365	0.9372	0.9732
20	0.8814	0.8801	0.8896	0.8911	0.9689
30	0.8312	0.8297	0.8448	0.8463	0.9508
40	0.7825	0.7809	0.7991	0.8016	0.9246

As shown in Table 5, the performance of FedAvg and methods that provide only cryptographic security decreases markedly as the proportion of malicious clients increases. When malicious clients account for 40%, the AUC of FedAvg drops to 0.7825, whereas DT-VPFL still achieves 0.9246. Secure aggregation primarily protects model-update privacy and cannot prevent legitimately authenticated clients from intentionally submitting malicious parameters. DT-VPFL continuously reduces the contribution of anomalous updates through dynamic trust selection and trust weighting. Figure 4 illustrates the performance trends under different proportions of malicious clients.

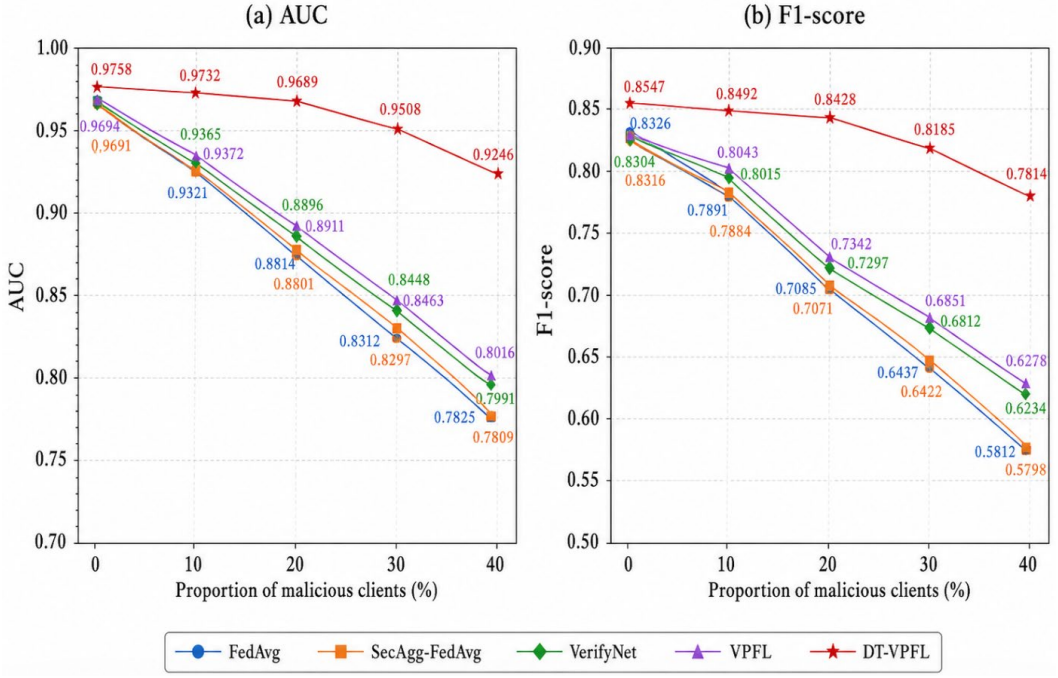

Figure 4. Model robustness under different proportions of malicious clients

4.3.2 Dynamic Trust Evolution

To evaluate the discriminative capability of the dynamic trust mechanism, representative normal, low-quality, and malicious clients are selected, and their trust values are recorded over 100 training rounds, as shown in Figure 5.

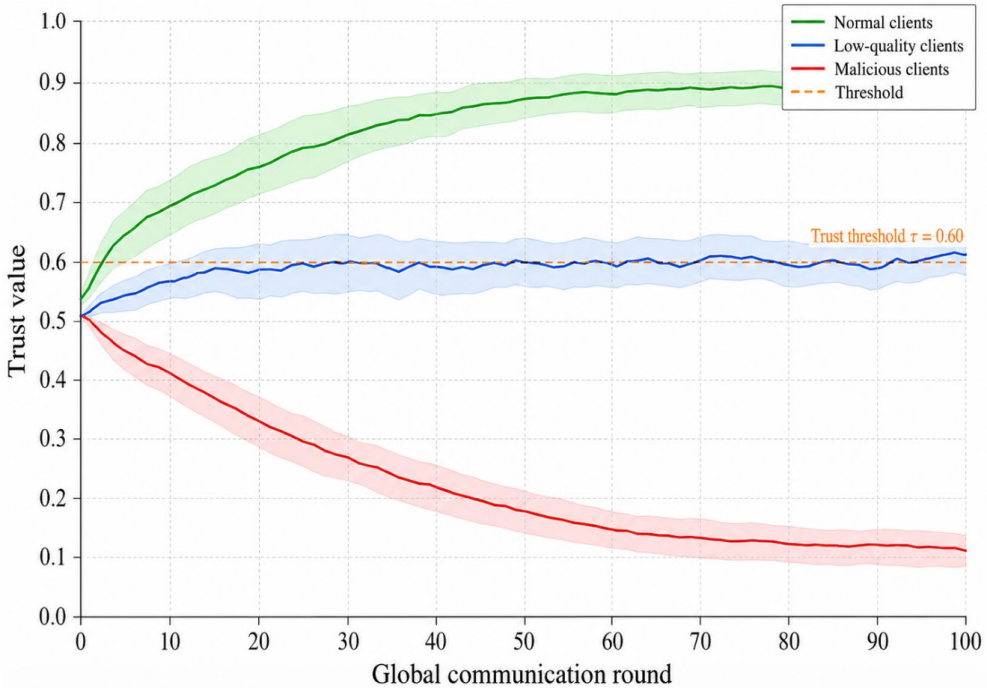

Figure 5. Evolution of dynamic trust values for different client types

Normal clients continuously perform effective training and pass integrity verification, and their trust values gradually stabilize at approximately 0.85-0.92. Low-quality clients with strong data heterogeneity or temporary degradation in training quality fluctuate mainly within 0.55-0.65 and remain near the trust threshold. Malicious clients gradually fall below 0.30 because of abnormal update directions or deteriorating verification records and ultimately drop below the trust threshold ($\tau = 0.60$). The malicious-client detection rate is defined as:

$$\mathrm{DR} = \frac{N_{\mathrm{detected}}^{\mathrm{mal}}}{N_{\mathrm{mal}}} \tag{25}$$

When the proportion of malicious clients is 20%, DT-VPFL achieves an average DR of 0.956 and a normal-client FER of 0.038, indicating that the dynamic trust mechanism can effectively distinguish persistently abnormal clients while avoiding excessive exclusion of normal nodes due to single-round fluctuations.

4.4. Privacy Preservation and Verifiability

4.4.1 Model Update Privacy Protection

To evaluate privacy protection for model updates, model-update inversion attacks are performed against FedAvg and DT-VPFL. In FedAvg, the attacker can directly obtain the model update uploaded by an individual client, whereas DT-VPFL exposes only masked parameters.

The normalized mean square error (NMSE) is used to measure the difference between the true load sequence x and the reconstructed sequence $\hat{x}$:

$$\mathrm{NMSE} = \frac{|x - \hat{x}|_2^2}{|x|_2^2} \tag{26}$$

The experimental results show that FedAvg has an average NMSE of 0.148 and a Pearson correlation coefficient of 0.872 between the reconstructed and original sequences. For DT-VPFL, the NMSE increases to 0.932 and the correlation coefficient decreases to 0.091. These results indicate that random masking and threshold secret sharing substantially impair the attacker's ability to recover user load characteristics from an individual model update. The corresponding visualization is shown in Figure 6(a).

4.4.2 Verifiability of Data and Aggregation Results

Tampering with client secret shares, masked model updates, and server aggregation results is further simulated at tamper ratios of 5%, 10%, 20%, and 30%. The verification detection rate is defined as:

$$\mathrm{VDR} = \frac{N_{\mathrm{detected}}^{\mathrm{tamper}}}{N_{\mathrm{tampered}}} \tag{27}$$

The corresponding VDR values reach 0.997, 0.998, 0.998, and 0.999, respectively, with an average false-positive rate of 0.003. Verifiable secret sharing detects incorrect secret shares, digital signatures detect forged or tampered uploads, and homomorphic-hash consistency verification detects incorrect aggregation results returned by the server. Figure 6(b) shows the reported verification results under different tamper ratios. The nonzero miss and false-positive rates are empirical workflow outcomes, not cryptographic collision probabilities. Exact integer verification should accept an untampered, consistently encoded aggregate; attribution of these observed errors requires event-level logs and a fully specified quantization implementation.

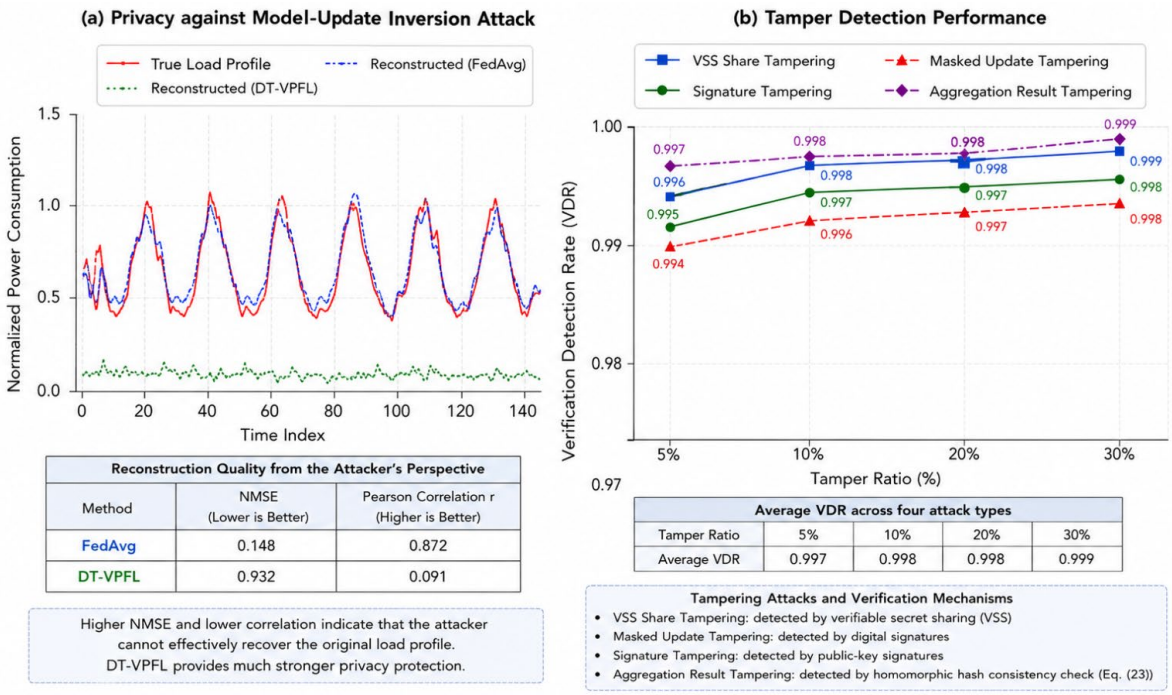

Figure 6. Privacy-preserving and verifiability results of DT-VPFL

Note: Figure 6(a) compares model-update inversion-attack results, while Figure 6(b) shows verification detection rates under different tamper ratios. The results demonstrate that DT-VPFL can simultaneously protect client-update privacy and provide two-layer verification for client inputs and server aggregation outputs.

4.5 Computational and Communication Overhead

Privacy-preserving and verifiable mechanisms introduce additional computational and communication costs. To evaluate the scalability of DT-VPFL, the total number of clients is set to $N \in \{10, 20, 40, 60, 80, 100\}$ while maintaining a 50% participation ratio per round. The average per-round training time and communication volume are recorded. When the numbers of clients are 10, 20, 40, 60, 80 and 100, the average per-round training times of DT-VPFL are approximately 2.98, 3.71, 5.26, 6.82, 8.37 and 9.95 s, respectively; the corresponding total communication volumes per round are approximately 12.4, 25.9, 56.8, 93.6, 136.9 and 186.7 MB.

As shown in Figure 7, computation time increases approximately linearly with the number of clients, while communication volume grows faster because secret shares must be exchanged among clients.

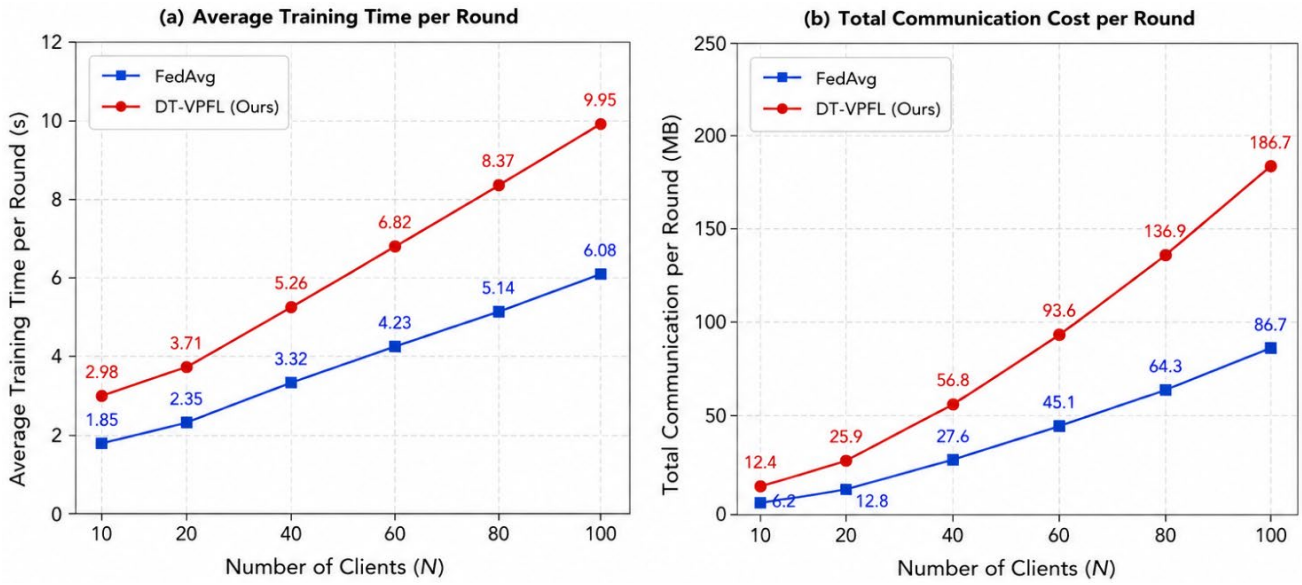

Figure 7. Computational and communication overhead under different client scales

Compared with FedAvg, DT-VPFL introduces secret sharing, signature verification, and aggregation-result verification, and therefore necessarily incurs additional runtime overhead. These operations are performed on model parameters and security-verification information rather than raw power data. Under the baseline setting with 20 clients, the reported average is 3.71 s per round. This original measurement is retained but its cryptographic coverage cannot be reconstructed from the available training description. The independently implemented protocol benchmarks in Section 4.8 use different dimensions and a separate environment; their costs must not be added to, substituted for, or described as already included in 3.71 s. Device measurements and an integrated training run are still needed.

For illustration, cohorts of m = 100 and 1,000 require 9,900 and 999,000 directed peer-share messages, respectively, approximately a 101-fold increase. These are combinatorial counts, not measured latency or bandwidth. They scale with the admitted cohort m rather than the registered population N. Bounded cohorts, stable edge gateways, and precomputed authentication material may reduce operational pressure. Hierarchical or sparse sharing could further reduce traffic but changes the collusion and verification model and has not been implemented or benchmarked here. Field deployment must budget uplink capacity, dropout retries, key management, energy use, and secure-scalar computation before selecting a cohort size.

4.6 Ablation Study

To further analyze the contribution of each core mechanism, the ablation study is conducted under 20% malicious clients and α_D=0.5, using three DT-VPFL variants: w/o Trust removes dynamic trust evaluation so that all legitimately authenticated clients participate in aggregation; w/o Trust Weight retains trusted-client selection but removes dynamic trust weighting; w/o Verification removes VSS, signatures, and aggregation-result verification while retaining dynamic trust and secure masking. The corresponding ablation results are shown in Table 6.

Table 6. Ablation results of DT-VPFL

Method	AUC	F1-score	DR	VDR	Per-round time / s
w/o Trust	0.9023±0.0078	0.7086±0.0104	0.000	0.996	3.21±0.08
w/o Trust Weight	0.9507±0.0049	0.8097±0.0081	0.928	0.997	3.52±0.10
w/o Verification	0.9586±0.0043	0.8254±0.0076	0.914	0.000	3.38±0.09
DT-VPFL	0.9689±0.0031	0.8428±0.0062	0.956	0.998	3.71±0.11

As shown in Table 6, removing dynamic trust reduces AUC from 0.9689 to 0.9023, indicating that cryptographic security mechanisms alone cannot resist poisoning updates generated by legitimately authenticated clients. Retaining trust-based selection while removing trust weighting decreases AUC to 0.9507, demonstrating that continuous trust weighting further limits the contribution of marginally low-trust clients.

After removing the verification mechanism, the model still achieves an AUC of 0.9586, but VDR drops to 0. This shows that dynamic trust can identify some model anomalies but cannot replace cryptographic verification of secret shares, communication data, and aggregation-result integrity. The complete DT-VPFL achieves an AUC of 0.9689, a malicious-client detection rate of 0.956, and a tamper detection rate of 0.998, confirming the complementary roles of dynamic trust, trust weighting, and verifiable secure aggregation.

Overall, DT-VPFL maintains model performance comparable to or better than conventional federated learning in attack-free settings and exhibits greater stability under strong Non-IID conditions and malicious-client participation. Random masking and threshold secret sharing effectively protect individual client model updates, while verifiable secret sharing, digital signatures, and aggregation-result verification detect active tampering during communication and aggregation. The measured range extends only to 100 registered clients with 50% scheduled participation. The results therefore support evaluation at moderate scale; they do not establish thousand-device scalability, field readiness, or end-to-end malicious security.

4.7 Analytical Sensitivity and Practical Parameter Selection

The following analysis follows directly from Eqs. (7), (12)–(14); it does not represent additional SGCC training runs. With scores in [0,1], transferring a weight ε from one trust component to another changes the instantaneous score by at most |ε|. With the previous state fixed, smoothing limits the current-round trust change to $(\mathbf{1}-\boldsymbol{\rho})|\boldsymbol{\varepsilon}|$. Thus, a 0.05 weight transfer at ρ = 0.80 changes current trust by at most 0.01. Under repeated perturbations the accumulated change can approach 0.05, so this one-round bound is not a long-run robustness guarantee.

After an isolated perturbation, its residual contribution decays geometrically with ρ. The half-lives for ρ = 0.60, 0.80, and 0.90 are approximately 1.36, 3.11, and 6.58 participating updates, respectively. Larger ρ provides greater smoothing but delays adaptation to a newly compromised client. At fixed trust values, increasing τ from 0.60 to 0.65 changes admission only for clients in [0.60,0.65); therefore, sensitivity depends on how many benign and malicious clients lie near the threshold. The low-quality-client interval reported in Figure 5 illustrates this trade-off, but does not establish an optimal τ.

For two admitted clients with equal sample counts and trust scores 0.9 and 0.6, the weight ratio is 1.5, 2.25, or 3.375 for η = 1, 2, or 3, respectively. A higher η amplifies trust differences but can concentrate influence and suppress useful heterogeneous clients. Together, these calculations explain parameter effects without asserting that the reported AUC is insensitive to them.

A prospective tuning procedure should reserve validation users, keep the test set untouched, and compare τ in {0.50,0.60,0.70}, ρ in {0.60,0.80,0.90}, and η in {1,2,3}. Each component weight can be varied by ±0.05 around Table 2 while redistributing the opposite change over the remaining nonnegative weights. Both clean and poisoning validation conditions should be examined using AUC/F1, malicious-client detection, benign exclusion, retained cohort size, and runtime. Select a setting subject to application-specific exclusion and availability limits, then confirm it with multiple fixed seeds and joint perturbations. This sweep is proposed for reproducibility and field calibration; it has not been executed in this revision. The ablation results in Table 6 are component comparisons, not a substitute for a hyperparameter sweep.

4.8 Protocol Implementation and Quantization Benchmarks

These supplementary experiments evaluate newly supplied protocol code on synthetic vectors. Runs used Linux x86-64, an Intel Xeon Platinum 8573C host, an 8-CPU quota and 8 GiB container memory, Python 3.12.14, NumPy 2.3.5, MP-SPDZ 0.4.3, and libsodium 1.0.18. Timings are from a shared container and three independent process executions or fresh-key trials per configuration, reported as mean ± sample standard deviation.

Secure consistency score. For d ∈ {64,256,1024}, seeds 0–2 generate normal inputs with standard deviation 0.05, clipped to [−1,1] for fixture generation and rounded to Q16. The public reference coordinate is round(65536 × 0.05 sin(j + 1))/65536 for j = 0,…,d − 1. Table 7 compares Eq. (9) on these quantized inputs with its float64 reference. The reported MP-SPDZ VM time includes fresh preprocessing. Data are its global data-sent counters in decimal MB; they are not a packet-capture measure of TLS/IP traffic. Compilation and certificate creation are excluded.

Table 7. Standalone MASCOT consistency scoring

d	VM time (s)	Sent (MB)	Max error
64	7.04 ± 0.27	2069.13	1.63e-10
256	9.65 ± 0.26	2690.73	1.63e-10
1024	17.30 ± 0.95	4956.26	1.55e-10

Across the nine random-vector executions, maximum absolute score error was 1.63×10^{-10}. Process wall-time

means were 7.20, 9.80, and 17.46 s. At d = 64, zero, opposite-reference, and aligned-reference controls returned 0.5, 0.00009444, and 0.99990556; ε explains the latter two values. A malformed bit value of 2 caused an explicit abort. Successful bit-validity openings were zero. The measured 2,069–4,956 MB per invocation demonstrate that a scalar output is not a low-communication protocol. This cold, unoptimized profile includes default preprocessing batches; no amortized or online-only performance is claimed.

Quantization accuracy. We tested $f \in \{12,16,20\}$, $s = 2^f$, and $m \in \{10,100,1000\}$ at d = 256, with seeds 0–19 for each configuration. Client coordinates follow $N(0,0.05^2)$; positive weights are sampled uniformly from [0.2,1] and normalized to sum to one. Table 8 reports the largest absolute coordinate error over 20 trials against the unquantized weighted sum. Every one of the 180 aggregates satisfied m/(2s). The m = 1,000 rows test arithmetic only, not a thousand-client cryptographic deployment.

Table 8. Quantization error over 20 trials per setting

f	m	Max error	Bound
12	10	8.570e-04	1.221e-03
12	100	2.661e-03	1.221e-02
12	1000	5.771e-03	1.221e-01
16	10	5.144e-05	7.629e-05
16	100	1.698e-04	7.629e-04
16	1000	4.975e-04	7.629e-03
20	10	3.303e-06	4.768e-06
20	100	9.718e-06	4.768e-05
20	1000	3.229e-05	4.768e-04

For the default f = 16, maxima at m = 10, 100, and 1,000 were 5.14×10^{-5}, 1.70×10^{-4}, and 4.97×10^{-4}. Signed tie cases confirmed nearest-even rounding. Nonfinite and out-of-bound coordinates were rejected, and a deliberately undersized test modulus failed the wraparound guard. Separately, 100 independently generated encoded aggregates at m = 10 and d = 64 all satisfied hash equality; changing one aggregate coordinate by one integer unit was rejected in all 100 trials. This is a functional check, not an estimate of a cryptographic collision probability or a guarantee of unchanged learning accuracy.

Masked aggregation implementation. Table 9 measures a serial functional harness that holds all participants in one process, invokes real cryptographic primitives, and serializes the logical protocol records. It includes fresh keys and masks, signed roster delivery, coefficient commitments, peer encryption, share verification, reconstruction, signed hash dissemination, and each recipient's aggregate check. Training, secure scoring, network transport, enrollment infrastructure, and negative controls are excluded. Logical MB count application records and their recipients, including naive commitment dissemination; they exclude transport overhead. Therefore, these are reference-harness costs, not deployed federated round latency.

Table 9. Serial aggregation harness with f = 16

m	d	k	Total (s)	Logical MB
5	64	3	0.54 ± 0.01	0.231
10	64	6	3.47 ± 0.01	1.502
20	64	12	25.04 ± 0.12	10.848
10	256	6	13.78 ± 0.06	5.901

At m = 10, d = 64, and k = 6, the mean time was 3.47 s, of which peer encryption and share checks consumed 2.85 s. Direct coordinatewise Feldman verification dominates this profile and is expensive as m grows. In all 12 trials, reconstruction with exactly k surviving holders recovered the exact encoded sum after every update had been uploaded. All six negative controls in each trial passed: changed aggregate, forged share, modified ciphertext, wrong session context, invalid signature, and fewer than k holders. These 72 controls test the specified failure paths; they do not constitute a malicious-composition proof or a real-network churn evaluation.

5. Conclusion

This paper proposes DT-VPFL, a dynamic trust-driven verifiable privacy-preserving federated learning method for smart grids. Client trust is continuously updated according to training quality, update consistency, historical behavior, and verification status, while trusted-client selection and trust weighting are used to suppress malicious updates. Random masking, threshold secret sharing, digital signatures, and homomorphic hashing are combined to hide local updates and verify aggregation results. Experiments on the SGCC dataset show that, in the attack-free setting, ACC, F1-score, and AUC reach 0.9609, 0.8547, and 0.9758, respectively. Under strongly Non-IID data, the AUC remains 0.9617, exceeding FedAvg by 0.0181. When 40% of clients are malicious, DT-VPFL still achieves an AUC of 0.9246 compared with 0.7825 for FedAvg. With 20% malicious clients, the detection rate reaches 0.956. Under model inversion attacks, NMSE increases from 0.148 to 0.932 and the Pearson correlation decreases from 0.872 to 0.091, while the tamper detection rate reaches 0.997-0.999. With 20 clients, the average training time is 3.71 s per round. These results support the proposed trust–verification integration within the reported experimental setting. The added sensitivity analysis is analytical, not a new training sweep. The supplementary

implementation specifies secure scoring and integer-domain aggregation. At d = 64, 256, and 1,024, cold MASCOT execution takes 7.04, 9.65, and 17.30 s on average, with maximum score error 1.63×10^{-10}. At s = 65,536 and m = 1,000, the largest tested aggregate-coordinate error is 4.97×10^{-4}; all 100 single-coordinate tamper controls are rejected. These independent component tests reveal substantial cryptographic costs and do not reproduce the SGCC training results. Cross-protocol input binding, leakage from public hashes, integrated training costs, and real-network churn remain deployment limitations.

References

[1] McMahan B, Moore E, Ramage D, Hampson S, Agüera y Arcas B. Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. PMLR, 2017, 54: 1273–1282.

[2] Yang Q, Liu Y, Chen T, Tong Y. Federated Machine Learning: Concept and Applications. ACM Transactions on Intelligent Systems and Technology, 2019, 10(2): Article 12, 1–19. DOI: 10.1145/3298981.

[3] Zheng R, Sumper A, Aragüés-Peñalba M, Galceran-Arellano S. Advancing Power System Services With Privacy-Preserving Federated Learning Techniques: A Review. IEEE Access, 2024, 12: 76753–76780. DOI: 10.1109/ACCESS.2024.3407121.

[4] Asghar M R, Dán G, Miorandi D, Chlamtac I. Smart Meter Data Privacy: A Survey. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2820–2835. DOI: 10.1109/COMST.2017.2720195.

[5] Sarmento E M D M, Ribeiro I F, Neris Y G, Rocha H R O, Mota V F S, Reginato C. Forecasting Energy Power Consumption Using Federated Learning in Edge Computing Devices. Internet of Things, 2024, 25: 101050. DOI: 10.1016/j.iot.2023.101050.

[6] Lin W T, Chen G, Zhou X. Privacy-Preserving Federated Learning for Detecting False Data Injection Attacks on Power System. Electric Power Systems Research, 2024, 229: 110150. DOI: 10.1016/j.epsr.2024.110150.

[7] Deng S, Zhang L, Yue D. Data-Driven and Privacy-Preserving Risk Assessment Method Based on Federated Learning for Smart Grids. Communications Engineering, 2024, 3: 154. DOI: 10.1038/s44172-024-00300-6.

[8] Alshamasi R Z, Ibrahim D M. Federated Intelligence for Smart Grids: A Comprehensive Review of Security and Privacy Strategies. Journal of Electrical Systems and Information Technology, 2025, 12: 43. DOI: 10.1186/s43067-025-00235-8.

[9] Zhang P, Zhang Y, Wang Z, Zhang Q, Gu Z. Privacy-Preserving Federated Learning Scheme for Distributed Smart Grid Based on Multi-Key Homomorphic Encryption. Peer-to-Peer Networking and Applications, 2025, 18: 293. DOI: 10.1007/s12083-025-02116-3.

[10] Jerkovic F, Sarkar N I, Ali J. Smart Grid IoT Framework for Predicting Energy Consumption Using Federated Learning Homomorphic Encryption. Sensors, 2025, 25(12): 3700. DOI: 10.3390/s25123700.

[11] Xue J, Yan W. Edge Computing Communication Privacy Protection Method Based on Federated Learning Algorithm. EAI Endorsed Transactions on Scalable Information Systems, 2026, 12(9). DOI: 10.4108/eetsis.12254.

[12] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan H B, Patel S, Ramage D, Segal A, Seth K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017: 1175–1191. DOI: 10.1145/3133956.3133982.

[13] Xu R, Li B, Li C, Joshi J B D, Ma S, Li J. TAPFed: Threshold Secure Aggregation for Privacy-Preserving Federated Learning. IEEE Transactions on Dependable and Secure Computing, 2024, 21(5): 4309–4323. DOI: 10.1109/TDSC.2024.3350206.

[14] Guo Z, Sun Y, Zhang X, Wu L. An Efficient Privacy-Preserving Secure Aggregation Scheme for Federated Learning with Input Verification and Dropout Resistance. EAI Endorsed Transactions on Scalable Information Systems, 2026, 12(8). DOI: 10.4108/eetsis.11991.

[15] Zhang Y, Wang Y, Liu W, Han G, Cao J, Tian Y. Secure Aggregation with Verifiability and Robustness for Privacy-Preserving Federated Learning. Knowledge-Based Systems, 2026, 336: 115310. DOI: 10.1016/j.knosys.2026.115310.

[16] Zhang Y, Zeng D, Luo J, Fu X, Chen G, Xu Z, King I. A Survey of Trustworthy Federated Learning: Issues, Solutions, and Challenges. ACM Transactions on Intelligent Systems and Technology, 2024, 15(6): Article 112, 1–47. DOI: 10.1145/3678181.

[17] Zhu L, Liu Z, Han S. Deep Leakage from Gradients. In: Advances in Neural Information Processing Systems. 2019, 32: 14774–14784.

[18] Geiping J, Bauermeister H, Dröge H, Moeller M. Inverting Gradients—How Easy Is It to Break Privacy in Federated Learning? In: Advances in Neural Information Processing Systems. 2020, 33: 16937–16947.

[19] Xu G, Li H, Liu S, Yang K, Lin X. VerifyNet: Secure and Verifiable Federated Learning. IEEE Transactions on Information Forensics and Security, 2020, 15: 911–926. DOI: 10.1109/TIFS.2019.2929409.

[20] Hahn C, Kim H, Kim M, Hur J. VerSA: Verifiable Secure Aggregation for Cross-Device Federated Learning. IEEE Transactions on Dependable and Secure Computing, 2023, 20(1): 36–52. DOI: 10.1109/TDSC.2021.3126323.

[21] Fu A, Zhang X, Xiong N, Gao Y, Wang H, Zhang J. VFL: A Verifiable Federated Learning With Privacy-Preserving for Big Data in Industrial IoT. IEEE Transactions on Industrial Informatics, 2022, 18(5): 3316–3326. DOI: 10.1109/TII.2020.3036166.

[22] Wang Y, Zhang A, Wu S, Yu S. VOSA: Verifiable and Oblivious Secure Aggregation for Privacy-Preserving Federated Learning. IEEE Transactions on Dependable and Secure Computing, 2023, 20(5): 3601–3616. DOI: 10.1109/TDSC.2022.3226508.

[23] Wu J, Zhang W. On the Security of Verifiable and Oblivious Secure Aggregation for Privacy-Preserving Federated Learning. IEEE Transactions on Dependable

and Secure Computing, 2024, 21(5): 4324–4326. DOI: 10.1109/TDSC.2024.3352170.
[24] Shen X, Luo X, Yuan F, Wang B, Chen Y, Tang D, Gao L. Verifiable Privacy-Preserving Federated Learning Under Multiple Encrypted Keys. IEEE Internet of Things Journal, 2024, 11(2): 3430–3445. DOI: 10.1109/JIOT.2023.3296637.
[25] Wang R, Xiong L, Geng J, Xie C, Li R. An Effective and Verifiable Secure Aggregation Scheme with Privacy-Preserving for Federated Learning. Journal of Systems Architecture, 2025, 161: 103364. DOI: 10.1016/j.sysarc.2025.103364.
[26] Zhang L, Tang B, Xu J. Lightweight Verifiable Privacy Preserving Federated Learning. Journal of Network and Computer Applications, 2025, 244: 104335. DOI: 10.1016/j.jnca.2025.104335.
[27] Cheng G, Liu J. Intelligent and Secure Federated Learning for Data Elements Circulation in Heterogeneous Edge Computation. EAI Endorsed Transactions on Scalable Information Systems, 2026, 13(1). DOI: 10.4108/eetsis.14118.
[28] Shabbir A, Manzoor H U, Zoha A, Halim Z. Smart Grid Security Through Fusion-Enhanced Federated Learning Against Adversarial Attacks. Engineering Applications of Artificial Intelligence, 2025, 157: 111169. DOI: 10.1016/j.engappai.2025.111169.
[29] Shepherd P, Dagiuklas T, Alkan B, Rodriguez J. Dynamic Trust Management for Secure Federated Learning in Critical Industrial and IoT Networks. In: 2025 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, 2025: 371–376. DOI: 10.1109/ICCWorkshops67674.2025.11162455.
[30] Xiao Y, Jin X, Pan T, Yu Z, Ding L. A Federated Learning Algorithm That Combines DCScaffold and Differential Privacy for Load Prediction. Energies, 2025, 18(6): 1482. DOI: 10.3390/en18061482.
[31] Zheng Z, Yang Y, Niu X, Dai H N, Zhou Y. Wide and Deep Convolutional Neural Networks for Electricity-Theft Detection to Secure Smart Grids. IEEE Transactions on Industrial Informatics, 2018, 14(4): 1606–1615. DOI: 10.1109/TII.2017.2785963.
[32] Li Y, Liu T T, Ling H C, et al. A robust federated learning algorithm for partially trusted environments. Computers & Security, 2025, 148: 104161. DOI: 10.1016/j.cose.2024.104161.
[33] Xiao Y. Adaptive trust evaluation and representation-based robust aggregation against poisoning attacks in federated learning. Scientific Reports, 2026, 16: 19800. DOI: 10.1038/s41598-026-50985-2.
[34] Nielsen J B, Nordholt P S, Orlandi C, Burra S S. A New Approach to Practical Active-Secure Two-Party Computation. arXiv preprint, 2012, arXiv:1202.3052. DOI: 10.48550/arXiv.1202.3052.
[35] Keller M. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2020: 1575–1590. DOI: 10.1145/3372297.3417872.
[36] Keller M, Orsini E, Scholl P. MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 830–842. DOI: 10.1145/2976749.2978357.
[37] Libsodium. Ristretto. Official documentation. https://doc.libsodium.org/advanced/point-arithmetic/ristretto. Accessed 2 October 2026.