An Efficient Privacy-Enhancing Cross-Silo Federated Learning and Applications for False Data Injection Attack Detection in Smart Grids

Hong-Yen Tran, Jiankun Hu, *Senior Member, IEEE*, Xuefei Yin, and Hemanshu R. Pota

***Abstract*— Federated Learning is a prominent machine learning paradigm which helps tackle data privacy issues by allowing clients to store their raw data locally and transfer only their local model parameters to an aggregator server to collaboratively train a shared global model. However, federated learning is vulnerable to inference attacks from dishonest aggregators who can infer information about clients' training data from their model parameters. To deal with this issue, most of the proposed schemes in literature either require a non-colluded server setting, a trusted third-party to compute master secret keys or a secure multiparty computation protocol which is still inefficient over multiple iterations of computing an aggregation model. In this work, we propose an efficient cross-silo federated learning scheme with strong privacy preservation. By designing a double-layer encryption scheme which has no requirement to compute discrete logarithm, utilizing secret sharing only at the establishment phase and in the iterations when parties rejoin, and accelerating the computation performance via parallel computing, we achieve an efficient privacy-preserving federated learning protocol, which also allows clients to dropout and rejoin during the training process. The proposed scheme is demonstrated theoretically and empirically to provide provable privacy against an honest-but-curious aggregator server and simultaneously achieve desirable model utilities. The scheme is applied to false data injection attack detection (FDIA) in smart grids. This is a more secure cross-silo FDIA federated learning resilient to the local private data inference attacks than the existing works.**

***Index Terms*— Privacy-preserving, federated learning, encryption, secret sharing, false data injection attack detection.**

I. Introduction

FEDERATED learning [1] is an emerging machine learning paradigm which addresses critical data privacy issues by enabling clients to store their raw data locally and transfer only their updated local model parameters to an aggregator server for jointly training a global model. Due to this characteristic, federated learning offers significant privacy improvements over centralizing all the training data. However, federated learning is vulnerable to inference attacks from dishonest aggregators who can infer information about clients' training data from their model parameters (weights, gradients) [2], [3], [4], [5], [6], [7]. For example, [4] employed generative adversarial networks to infer the private data of a target client from its shared model parameters. This means that even if the model is trained in federated learning, data privacy still cannot be rigorously guaranteed. Information can be extracted from global model parameters, but this information cannot be linked to a specific single client because the data samples are anonymized among multiple clients. However, this is not the case if the information is inferred from local model parameters by a corrupted aggregator. Thus, clients' model parameters should be protected from the access of a corrupted aggregator to prohibit these potential inference attacks.

To address this problem, existing approaches focus on two main techniques, which are differential privacy-based and secure aggregation-based. The former adds noise directly to the client's models over a numerous number of iterations; thus, it has the drawbacks of sacrificing the global model accuracy to make a trade-off of privacy-utility. The latter utilizes techniques in cryptography such as secure multiparty computation and homomorphic encryption to securely aggregate the clients' models without knowing their specific values. However, most of these existing approaches rely on a trusted third party to generate the master key for aggregation or a setting with multiple non-colluding servers. Besides, many proposed schemes are still inefficient and impractical due to the expensive overhead of computation and communication among multiple clients over multiple rounds of training.

False data injection attack (FDIA) detection [8], [9] is a critical security operation in a smart grid control system. and has been solved by data-driven machine learning methods. The data-driven machine learning methods require a huge amount of measurement data which are distributed over an interconnected grid. In such an interconnected grid, each sub-grid is possessed and managed by an independent transmission grid company (TGC) regarding power industry deregulation [10], [11]. To build a high-accuracy model for false data injection detection, measurement data from all involved sub-grids

Manuscript received 15 April 2022; revised 22 February 2023; accepted 11 April 2023. Date of publication 17 April 2023; date of current version 26 April 2023. This work was supported in part by ARC Discovery Grant DP190103660 and Grant DP200103207, and in part by ARC Linkage Grant LP180100663. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Chia-Mu Yu. *(Corresponding author: Jiankun Hu.)*

Hong-Yen Tran, Jiankun Hu, and Hemanshu R. Pota are with the School of Engineering and Information Technology, The University of New South Wales Canberra at ADFA, Canberra, ACT 2602, Australia (e-mail: hongyen.tran@student.adfa.edu.au; j.hu@adfa.edu.au; h.pota@adfa.edu.au).

Xuefei Yin is with the School of Information and Communication Technology, Griffith University, Gold Coast, QLD 4222, Australia (e-mail: x.yin@griffith.edu.au).

Digital Object Identifier 10.1109/TIFS.2023.3267892

should be shared. However, transmitting such huge measurement data over the network for a centralized detection machine learning algorithm is expensive and also leads to security and privacy issues including competitive privacy [12]. The question is how to coordinate these TGCs to detect FDI attacks while preserving their competitive privacy. This remains a challenging problem which has been attracting recent studies with federated learning-based solutions. In federated learning, a cross-silo setting is often established where a number of companies or organizations have a common incentive to train a model based on all of their data, but do not share their data directly due to confidentiality/privacy or legal constraints [13]. To enhance the privacy of power companies when they contribute their local training models, an efficient privacy-preserving cross-silo federated learning for FDIA detection over multi-area transmission grids should be designed.

In view of the above issues, we propose an efficient cross-silo federated learning with strong privacy preservation which can be applicable to the smart grid domain. By designing a double-layer encryption scheme over multiple federated learning rounds and utilizing Shamir secret sharing, we achieve an efficient privacy-preserving federated learning protocol, which also allows some clients to drop out and rejoin dynamically during the training process. Specifically, we summarize the main contributions as follows:

- A general privacy-enhancing cross-silo federated learning with a secure weighted aggregation scheme is designed based on lightweight double-layer encryption and Shamir secret sharing. The scheme removes the requirement of computing discrete logarithms which is the limitation of some related works. No multiple non-colluding server settings are required. Besides, clients' secret keys of two encryption layers are generated in a decentralized manner which helps increase privacy.
- The proposed scheme is demonstrated theoretically and empirically to provide provable privacy against an honest-but-curious aggregator server and simultaneously achieve desirable model utility.
- The proposed scheme is efficient in communication/computation and robust against dropouts/rejoining during training iterations.
- An efficient privacy-enhancing cross-silo federated learning resilient to the local training data inference attacks for FDIA detection in the smart grid domain is proposed and empirically evaluated.

This paper consists of eight sections. Following this Introduction section are the Related Works and Preliminaries sections. The proposed privacy-enhancing cross-silo federated learning without any trusted third parties is given in Section IV, followed by the analysis of the scheme in Section V. A concrete scenario of enhancing privacy in cross-silo federated learning for FDIA detection in smart grids with empirical evaluation is given in Section VI and Section VII. Finally, Section VIII is for the discussion and conclusions.

II. Related Works

Existing works on enhancing privacy for federated learning mainly employ two types of techniques. One technique is differential privacy [14], which adds appropriate noise to shared parameters according to the desired privacy level. For example, [15] added Laplace noise to the gradients and selectively shared the perturbed gradients, [16], [17] presented a client-sided differential privacy federated learning scheme to hide clients' model contributions during training. To protect local models, the added noise to each local model must be big enough, resulting in the aggregate noise corresponding to the aggregate model being too large, which would completely destroy the utility of this model.

The other technique is secure multiparty computation and homomorphic encryption for secure aggregation. The scheme in [18] was based on Elgamal homomorphic encryption. This scheme requires a trusted dealer to provide each participant with a secret key sk_i and the aggregator sk_0 such that $\sum_{i=0}^{k} sk_i = 0$. Their private secure aggregation is aggregator oblivious in the encrypt-once random oracle model where each participant only encrypts once in each time period. To decrypt the sum, it ends up computing the discrete logarithm which can be implemented through a brute-force search or Pollard's lambda method which requires $O(\sqrt{k\Delta})$, where k is the number of parties and Δ is the maximum value of any party's input. To overcome the limitations of solving discrete logarithm problems, [19] presented a scheme in the encrypt-once random oracle model with fast encryption and decryption based on Decisional Composite Residuosity Assumption which removes the discrete logarithm computation. However, this scheme also requires a trusted dealer to generate and distribute the secret keys to participants and an aggregator. Besides, both of the approaches in [18] and [19] only deal with secure aggregation of scalars over periods of time (not the secure weighted aggregation of model vectors over multiple iterations of federated learning) and does not deal with dropouts/rejoining problems. Addressing the drawbacks of [18] and [19], the work in [20] proposed a secure aggregation scheme where the input is a vector and can deal with dropouts. The scheme is based on pairwise additive stream ciphers and Shamir secret sharing to tackle client failures. Diffie-Hellman key exchange is adopted to share common pair-wise seeds of a pseudorandom generator. Double-masking is introduced to prevent leakage if there is any delay in transmission. Nevertheless, this approach requires at least four communication rounds between each client and the aggregator in each iteration and a repetition of Shamir secret sharing for each iteration. Thus, it suffers from communication and computation inefficiency considering the huge number of iterations of federated learning. Utilizing the technique of secure data aggregation in [20], the work in [21] proposed a general privacy-enhanced federated learning scheme with secure weighted aggregation, which can deal with both the data significance evaluation and secure data aggregation. This scheme still inherits the same drawbacks as [20]. Besides, this scheme only resolved a weak security model where no collusion between the server and the clients participating in the federated learning. The paper [22] presented Prio, a privacy-preserving system for the collection of aggregate statistics. With a similar approach, [23] introduced SAFELearn, a generic design for efficient private federated learning systems that protect against inference attacks using

secure aggregation. However, these designs rely on multiple non-colluded server settings. Dong et. al. in [24] designed two secure ternary federated learning protocols against semi-honest adversaries based on threshold secret sharing and homomorphic encryption respectively. In the first protocol, threshold secret sharing is used to share all local gradient vectors in all iterations, which causes expensive computation and communication overhead. Besides, the limitation of their second protocol is that all clients use the same secret key and if the server colludes with a client then it can obtain all client's models. In [25], Fang et. al. modified the traditional ElGamal protocol into a double-key encryption version to design a new scheme for federated learning with privacy preservation in cloud computing. Nevertheless, the scheme has to solve the discrete logarithm problem as [18]. The study in [26] combined additively homomorphic encryption with differential privacy but cannot tolerate client dropouts. Their system creates significant run-time overheads which makes it impractical for real-world federated learning applications. Functional encryption and differential privacy is utilized in [27] to design the HybridAlpha scheme. However, HybridAlpha relies on a trusted party that holds the master keys. The proposed scheme in [28] replaced the complete communication graph in [20] with a k-regular graph of the logarithmic degree to reduce the communication cost while maintaining the security guarantees; however, each client shares its secret across only a subset of parties, and thus the dropout-resilience is downgraded.

Considering the integrity of the global model besides the privacy preservation of the local data and models, the proposed approach in [29] combined the Paillier additive homomorphic and verifiable computation primitives. The scheme in [29] can verify the correctness of the aggregated model given the fact that every client provides their genuine local models. From the perspective of privacy preservation, the scheme can only tolerate a weaker threat model. No collusion among the server and clients participating in the federated learning protocol was assumed as the keys (sk, pk) necessary for the homomorphic encryption and the signatures are generated by one of the clients and shared among all clients. In the work [17], to deal with the problem of collusion in [29], adding Gaussian noise to the local models before homomorphically encryption was proposed. However, the standard variation of the additive Gaussian noise must be small to not destroy the genuine local models, resulting in the fact that the adding noise protection is not able to provide a high level of differential privacy (ε is not small, i.e., less than 1).

The power grid scenario of false data injection attack detection based on federated learning in smart grids has been studied in [30], [31], and [32]. The investigated power grid scenario is similar in these papers and in the proposed scheme. For example, in [30] an independent power system state owner (PSSO) and a detection service provider (DSP) correspond to an independent transmission grid company (TGC) and a system operator (SO) in the proposed scheme. The power grid scenario fits with the investigated cross-silo federated learning setting (e.g., the number of parties (PSSOs/TGCs) is small and each party is facilitated with high-performance computing). However, [30] and [31] only apply federated learning and do not consider the security problem of local data privacy leakage from local models as in [32] and our proposed scheme. The scheme in [32] enhanced privacy by utilising Pallier-based homomorphic encryption for secure model aggregation, but only resolved a weak security model where no collusion among the server and the clients participating in the federated learning. All clients have to share a common pair of public key and secret key for encryption/decryption and a trusted party is required to generate this key pair.

A privacy-preserving federated learning approach needs to be efficient in computation and communication while providing strong privacy preservation and desirable model utility. Most of the related works focus on the basic problem of secure aggregation with the main approaches based on secure multi-party computation, homomorphic encryption, and differential privacy. In spite of some achievements in secure aggregation and privacy-preserving federated learning, there are still drawbacks. The majority of proposed schemes in literature either require a trusted third party to compute master secret keys or all local parties share a common secret key or non-colluded server settings. This means these works guarantee privacy in weaker security models (e.g., no collusion).

The proposed scheme does not require a trusted dealer to provide each participant with a secret key as the scheme in [18], [19], [27], and [32]. While the schemes in [18] and [25] require computing the discrete logarithm, our scheme removes that complexity by utilizing the encryption-decryption based on the Decisional Composite Residuosity assumption. Moreover, both of the approaches in [18] and [19] only deal with secure aggregation of scalars over periods of time, not the secure weighted aggregation of model vectors over multiple iterations of federated learning. The dropout and rejoining problems were not investigated in these works too. Although eliminating the drawbacks in [18] and [19], the schemes in [20] and [28] suffer higher computation overhead than the proposed approach and do not address federated learning with secure weighted aggregation. Other systems in [22] and [23] depend on multiple non-colluded server settings, which is not required with our scheme. The systems in [21], [24], [29], and [32] cannot tolerate the risk of revealing all clients' models when there is a collusion between the server and a client as our protocol. The study in [26] cannot resolve client dropouts. Their system creates significant run-time overheads, making it impractical for real-world federated learning applications. Our scheme is resilient to dropouts and provides efficient performance for real applications, such as privacy-preserving federated learning false data injection detection.

To summarize, Table I gives a comparison of our scheme with related works regarding the application scenario of FDIA federated learning with secure weighted aggregation (A1, A2) and different security/privacy properties (A3-A8). Only three recent works [30], [31], [32] studied the FDIA federated learning. Most of the related works do not provide all security properties A3-A8. Only the studies in [20] and [28] filtered from Table I satisfy all security/properties as the proposed approach. Table II compares the computation and communication complexity between these two studies [20], [28] and the proposed scheme. From Table I and Table II, it can be seen that

TABLE I
COMPARISON OF OUR SCHEME WITH RELATED WORKS. A1. FDIA FEDERATED LEARNING, A2. SECURE WEIGHTED MODEL AGGREGATION, A3. NO TRUSTED DEALER TO GENERATE AND DISTRIBUTE SECRET KEYS, A4. NO NON-COLLUDED SERVER SETTING, A5. COLLUSION RESISTANCE, A6. DROPOUTS/REJOINS HANDLING, A7. NO DISCRETE LOGARITHM PROBLEM SOLVING, A8. PRIVACY-SECURITY TRADE-OFF

Work	A1	A2	A3	A4	A5	A6	A7	A8
[14]	×	×	✓	✓	✓	×	✓	×
[15]	×	×	✓	✓	✓	×	✓	×
[16]	×	×	✓	✓	✓	×	✓	×
[17]	×	×	✓	✓	✓	×	✓	×
[18]	×	×	×	✓	✓	×	×	✓
[19]	×	×	×	✓	✓	×	✓	✓
[20]	×	×	✓	✓	✓	✓	✓	✓
[21]	×	✓	×	✓	×	✓	✓	✓
[22]	×	×	✓	×	×	×	✓	✓
[23]	×	×	✓	×	×	✓	✓	✓
[24]	×	×	×	✓	×	✓	✓	✓
[25]	×	×	✓	✓	✓	✓	×	✓
[26]	×	×	×	✓	✓	×	✓	✓
[27]	×	×	×	✓	✓	✓	✓	✓
[28]	×	×	✓	✓	✓	✓	✓	✓
[29]	×	×	×	✓	×	✓	✓	✓
[30]	✓	×	n/a	n/a	n/a	n/a	n/a	n/a
[31]	✓	×	n/a	n/a	n/a	n/a	n/a	n/a
[32]	✓	×	×	✓	×	✓	✓	✓
Our	✓	✓	✓	✓	✓	✓	✓	✓

the proposed scheme guarantees privacy in a stronger security model and at a lower computational overhead than the related works.

III. PRELIMINARIES

A. Notations and Definitions

Column vectors are denoted by lower-case bold letters, like $\boldsymbol{v}$. The i-th entry of the vector $\boldsymbol{v}$ is v_i. $\boldsymbol{v}^{\mathrm{T}}$ is the transpose of the column vector $\boldsymbol{v}$. The zero-vector is represented by $\mathbf{0}$. Given a set $\mathcal{S}$, $x \leftarrow_{\$} \mathcal{S}$ indicates that x is sampled uniformly at random from $\mathcal{S}$. The notion $[k]$ represents the set $\{0, 2, \ldots, k-1\}$. The computational indistinguishability of two distributions H_0 and H_1, is denoted by $H_0 \cong H_1$. Table III lists the notions used in this paper.

B. Federated Learning

Federated learning is a machine learning scheme where multiple clients collaborate in generating a shared machine learning model, under the coordination of a central server. Each client's raw data is stored locally and not transmitted; instead, their local model parameters are sent to the server for aggregation to achieve the learning objective. Cross-silo federated learning is the federated learning setting when clients are different organizations or geo-distributed data centres that have the incentive to train a shared model on the union of their siloed data. [13]

Several algorithms have been proposed for federated learning. In this work, we utilize FedAvg [1], which is the original federated learning aggregation mechanism and is commonly applied in related works. In FedAvg, the global model parameters are updated by summing the weighted local model parameters $\boldsymbol{w} = \sum_{i=1}^{k} \frac{n_i}{n} \cdot \boldsymbol{w}_i$.

C. Shamir Secret Sharing

(t, n) Shamir secret sharing scheme [33] creates k shares $\{s^{(1)}, \ldots, s^{(n)}\}$ of a secret s such that s can be efficiently reconstructed by any combination of t data pieces but cannot by any set of less than t data pieces.

$s, s^{(1)}, \ldots, s^{(n)}$ are the elements in a finite field $\mathbb{Z}_p$ for some large prime p where $0 < t \leq n < p$. The scheme works as follows:

- Setup: The secret holder randomly chooses $a_1, \ldots, a_{t-1}$ from $\mathbb{Z}_p$ and $a_0 = f(0) = s$ to define a polynomial of degree $t \leq 1$:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t\leq 1} x^{t\leq 1} \bmod p$$

- Sharing: The secret holder computes $s^{(i)} = f(i)$ for $i \in \{1, 2, \ldots, n\}$, and sends $(i, s^{(i)})$ to the corresponding participants i.
- Reconstructing: Given any t of $(i, s^{(i)})$ pairs, an user is able to reconstruct the secret

$$s = a_0 = \sum_{j=1}^{t} s^{(j)} \cdot \prod_{m=0, m\neq j}^{t} \frac{x_m}{x_m - x_j} \bmod p$$

D. Decisional Composite Residuosity Assumption

Let $N = p \cdot q$ for two large primes p and q. The Decisional Composite Residuosity (DCR) assumption [34] states that the advantage of a distinguisher $\mathcal{D}$, defined as the distance:

$$\mathsf{Adv}_{\mathcal{D}}^{\mathsf{DCR}} := |\mathsf{Pr}[\mathcal{D}(y, N) = 1 |\ y = x^N \bmod N^2, x \leftarrow_{\$} \mathbb{Z}_N^*] - \mathsf{Pr}[\mathcal{D}(y, N) = 1 |\ y \leftarrow_{\$} \mathbb{Z}_{N^2}^*]|$$

where probabilities are taken over all coin tosses, is a negligible function

E. False Data Injection Attacks

False data injection attacks (FDIAs) are designed by manipulating some measurements to circumvent the residual-based bad data detection in a power management system [8], [9], [35]. Various algorithms have been designed to detect these attacks using new techniques instead of the residual-based bad data detection mechanism. One example is the deep learning network to model the spatial-temporal relationship between bus and line measurements in [36].

TABLE II

COMPARISON OF SECURE AGGREGATION AMONG [20], [28] AND OURS. k IS THE NUMBER OF LOCAL PARTIES/CLIENTS. L IS THE LENGTH OF THE CLIENTS' MODEL VECTOR. τ IS THE SHAMIR SECRET SHARING THRESHOLD

		[20]	[28]	Ours
Computation overhead	Server	$O(L \cdot k^2)$	$O(L \cdot k \log k + k \log^2 k)$	$O(L + [k^2])$
	Client	$O(L \cdot k + k^2)$	$O(L \cdot \log k + \log^2 k)$	$O(L + [\tau \cdot k])$
Communication overhead	Server	$O(L \cdot k + k^2)$	$O(L \cdot k + k \log^2 k)$	$O(L \cdot k + [k^2])$
	Client	$O(L + k)$	$O(L + \log^2 k)$	$O(L + [k])$

TABLE III

LIST OF NOTATIONS

Notation	Description
k	The number of local parties
T	The number of learning iterations
$\mathcal{C}$	Set of controlled parties
$\mathcal{N}$	Learning network structure
$\mathbf{w}^{(t)}$	The global learning model vector at the t-th iteration
$\mathbf{x}_i^{(t)}$	The encoded local learning model vector of P_i at the t-th iteration
L	The length of the model vector
$[L]$	$= [0, \cdots, L-1]$
$\mathbb{Z}_N$	The additive group of integers modulo N
$\mathbb{Z}_N^*$	The multiplicative group of integers modulo N
$\mathcal{U} = \{\mathrm{P}_i\}_{i \in [k]}$	The set of all local parities' indices
$\mathcal{U}_a^t$	The set of alive parties' indices at the t-th iteration
$\mathcal{U}_d^t = \mathcal{U} \setminus \mathcal{U}_a^t$	The set of dropped parties' indices at the t-th iteration
$\mathcal{U}_r^t$	The set of parties' indices whose rejoins the t-th iteration
$s_i^{(j)}$	The share of secret s_i for P_j
$n_i^{(t)}$	The number of training data of P_i at iteration t-th
$n^{(t)}$	$= \sum_{i \in [k]} n_i^{(t)}$

IV. PROPOSED PRIVACY-ENHANCING CROSS-SILO FEDERATED LEARNING

A. System Model and Overview of the Proposed Privacy-Enhancing Cross-Silo Federated Learning

Consider a system with k local parties and an aggregator server. Each local party owns its private dataset $\mathcal{D}_i$, $i \in \{1, \ldots, k\}$ with $n_i = |\mathcal{D}_i|$ samples. All local participants agree on the same learning network structure $\mathcal{N}$. The global learning network model at t-th iteration consists of L weight parameters, denoted as $\boldsymbol{w}_G^{(t)} = \{w_0^{(t)}, w_1^{(t)}, \ldots, w_{L-1}^{(t)}\}$. The aim is to learn a global network model from all local datasets without exposing participants' data privacy under the coordination of the aggregator.

The adversary is the honest-but-curious aggregator server which is assumed to follow the protocol honestly, but attempts to infer sensitive information about participants' training data from their model updates $\boldsymbol{w}_i^{(t)}$. It is also assumed that there are private and authenticated peer-to-peer channels between parties so that the data transferred cannot be eavesdropped on or modified. This can be enforced in practice with the appropriate use of Digital Signatures and Certificate Authorities. To implement federated learning which utilizes the union of local datasets, for each iteration, each party contributes its local model vector $\boldsymbol{w}_i^{(t)}$. Unfortunately, this raises the risk of inference attacks performed by an honest-but-curious aggregator on each local model to extract information about the corresponding party's local data used for training. Hence, ordinary federated learning needs to be integrated with privacy protection techniques to prohibit access to individual model updates. The system should be designed in a way to hide local models from the aggregator to counter the inference attacks while still enabling efficient and accurate federated learning.

The following section introduces and explains the main techniques in the proposed privacy-enhancing cross-silo federated learning scheme.

B. High-Level Technical Overview

1) Protecting Local Models: The encryption scheme in [19] based on the DCR assumption in the random oracle model is utilized to obtain the global model vector as the weighted average function of a set of local model vectors given their encryptions $c_{i,j}^{(t)} = (1 + N_1)^{x_{i,j}^{(t)}} \cdot \mathcal{H}_1(j)^{sk_i^{(t)}} \mod N_1^2$. Here, $x_{i,j}^{(t)}$ is the j-th element of the i-th party's model vector encoded in a non-negative integer form at the t-th iteration, $sk_i^{(t)}$ is the secret encryption key of i-th party at the t-th iteration. The main benefit of this construction is that the weighted average global model vector can be retrieved without computing the discrete logarithm as the other approaches in literature [18], [25]. In [19], only the *secure aggregation* is considered and it is assumed that there exists a trusted dealer generates encryption key $sk_i, i = 1 \cdots k$ and the master key $sk_0 = -\sum_{i=1}^{k} sk_i$. In our proposed scheme, the *secure weighted aggregation* is investigated, each party creates its own secret key $sk_i^{(t)}$ and the master key is computed from clients' secret keys in a secure computation manner. To enable the secure weighted aggregation of local models which was not considered in [19], the number of each party's training samples is also encrypted by the corresponding $sk_i^{(t)}$ at each iteration. The master key to decrypt the global model vector is calculated as $msk^{(t)} = \sum_{i \in \mathcal{U}_a^t} sk_i^{(t)}$, where $\mathcal{U}_a^t$ is the set of alive parties who contribute their encrypted local models for aggregation. This master key should be computed in a secure way to increase privacy level. This is achieved by designing the second layer of the basic encryption scheme to encrypt the secret encryption keys $sk_i^{(t)}$ of the first layer, which is $\beta_i^{(t)} = (1 + N_2)^{sk_i^{(t)}} \cdot \mathcal{H}_2(t)^{v_i^{(t)}} \mod N_2^2$. The secret encryption key of this second layer is $v_i^{(t)}$. The requirement for $v_i^{(t)}$ is that it is privately generated by each party such that $\sum_{i \in \mathcal{U}} v_i^{(t)} = 0$, where $\mathcal{U}$ is the set of all parties. Different from the secret keys $\boldsymbol{sk}$ of the first layer which are generated at each iteration, the

secret encryption keys $\boldsymbol{v}^{(0)}$ which are created at the initial sub-protocol π_0 of the establishment phase basically can be used for multiple iterations ($\boldsymbol{v}^{(t)} = \boldsymbol{v}^{(t-1)} = \cdots = \boldsymbol{v}^{(0)}$). The generation of $\boldsymbol{v}^{(t)}$ is based on the correlated antiparticles using common pair-wise secrets, $v_i^{(t)} = \sum_{i<j} \gamma_{i,j}^{(t)} - \sum_{i>j} \gamma_{j,i}^{(t)}$, where $\gamma_{i,j}^{(t)}$ is the common initial pair-wise secrets between party i and party j created by adopting the Diffie-Hellman Key exchange protocol.

2) Handling Dropouts: Shamir's τ-out-of-k secret sharing is utilized to allow a user to split a secret into k shares, such that any τ shares can be used to reconstruct the secret, but any set of at most $\tau \leq 1$ shares gives no information. Each party creates k shares of its secret $s_i^{(t)}$, keeps one share and sends each share to each different party from $k-1$ remaining shares. At each iteration t, after receiving the ciphertexts, the aggregator broadcasts the set of alive parties $\mathcal{U}_a^t$, the set of the dropped parties $\mathcal{U}_d^t = \mathcal{U} \setminus \mathcal{U}_a^t$. If $\mathcal{U}_a^t = \mathcal{U}$ then we have $\sum_{i \in \mathcal{U}_a^t} v_i^{(t)} = 0$; but, if $\mathcal{U}_a^t \subset \mathcal{U}$ then the sum $\sum_{i \in \mathcal{U}_a^t} v_i^{(t)}$ needs to be recovered. Alive parties send their shares $s_d^{(ti)}$ of a dropped party P_d to the aggregator. Thanks to the τ-out-of-k Shamir threshold secret sharing scheme, the share $s_d^{(t)}$ can be recovered by the aggregator as long as the aggregator receives at least τ secret shares $s_d^{(ti)}$. Having $s_d^{(t)}$, the aggregator can compute $v_d^{(t)}$ and obtain $\sum_{i \in \mathcal{U}_d^t} v_i^{(t)}$ as the master key of the second encryption layer to obtain the sum $\sum_{j \in \mathcal{U}_a^t} sk_j^{(t)}$. Because the sum $\sum_{j \in \mathcal{U}_a^t} sk_j^{(t)}$ is the master key of the first encryption layer; thus, it helps to get the sum $\sum_{j \in \mathcal{U}_a^t} x_j^{(t)}$.

3) Handling Rejoining: Assume that the secret $s_d^{(t-1)}$ of a dropped party P_d was revealed to the aggregator. If P_d rejoins the current iteration, which is the t-th iteration, P_d has to create a new secret $s_d^{(t)}$. For this case, the party P_d needs to send its updated public key $pk_d = g^{s_d^{(t)}}$ to the aggregator, then creates and shares Shamir's shares of its updated secret $s_d^{(t)}$. The aggregator broadcasts the updated set of public keys and the set of rejoining parties. Rejoining parties P_r update the seeds $s_{r,i}^{(t)} = (pk_i^{(t)})^{s_r^{(t)}}$ shared with all other parties and compute their updated secret $v_r^{(t)}$. Other parties P_i update the seeds $s_{i,r}^{(t)} = (pk_r^{(t)})^{s_i^{(t)}}$ shared with the rejoining parties and also calculate their updated secret $v_i^{(t)}$.

4) Reducing Communication and Computation Overhead: To overcome the problem of communication and computation overhead in federated learning with multiple iterations, the proposed solution is threefold. The first one is to utilize a lightweight encryption/decryption scheme which has no requirement to compute discrete logarithms. The second one is to accelerate the computation performance via parallel computing of Single Instruction Multiple Data (SIMD) of cryptographic operations over model vectors and pre-computed hash functions. The third one is to limit the number of times of creating and transmitting the secrets $s_i^{(t)}$ in the Shamir secret sharing scheme. This is effectively performed by designing a double-layer encryption scheme where the secret keys $\boldsymbol{sk}$ of the first layer are used for only one iteration and the secret keys $\boldsymbol{v}$ of the second layer can be used for multiple iterations. Shamir's secret sharing for the secrets $\boldsymbol{s}$ is only implemented at the establishment phase and in the iterations when parties rejoin. Besides, only rejoining parties P_r generate new key pairs and transmit their new public keys $pk_r^{(t)} = g^{s_r^{(t)}}$.

C. Description of the Proposed Protocol

Algorithm 1 describes the overall steps of the proposed privacy-enhancing cross-silo federated learning from the client side and the server side.

Algorithm 1 Proposed Privacy-Enhancing Cross-Silo Federated Learning Algorithm

Input:

T: Maximum number of rounds, k: the number of clients selected in each round, N_{epoch}: the number of local epochs, and η: the local learning rate, pp: public parameters

Output:

Global model $\boldsymbol{w}_G$

Processing:

[*Server-side*]

Initialize $\boldsymbol{w}_G^0$
for each round t from 1 to T **do**
  $\mathcal{U}_t$ contains k clients
  for each client $i \in \mathcal{U}_t$ **in parallel do**
    $C_i^{(t)} \leftarrow$ **LocalTraining**$(i, \boldsymbol{w}_G^{(t)}, t)$
  end for
  $\mathbf{y}^{(t+1)}, n^{(t)} \leftarrow \mathsf{Dec}(pp, \{C_i^{(t)}\}_{i \in \mathcal{U}_a^t})$
  $\boldsymbol{w}_G^{(t+1)} = \frac{1}{n^{(t)}} \cdot \mathsf{Decode}(\mathbf{y}^{(t+1)})$
end for

[*Client-side*: Party P_i]

LocalTraining$(i, \boldsymbol{w}, t)$:

Divide local dataset $\mathcal{D}_i^{(t)}$ for round t into batches; $\mathcal{B}_i^{(t)}$ denotes the set of the batches.
for each epoch j from 1 to N_{epoch} **do**
  for each batch $b \in \mathcal{B}_i^{(t)}$ **do**
    $\boldsymbol{w}_i^{(t)} \leftarrow \boldsymbol{w}_i^{(t)} - \eta \nabla L(\boldsymbol{w}_i^{(t)}; b)$
  end for
end for
$\boldsymbol{z}_i^{(t)} \leftarrow n_i^{(t)} \cdot \boldsymbol{w}_i^{(t)}$
$\boldsymbol{x}_i^{(t)} \leftarrow \mathsf{Encode}(\boldsymbol{z}_i^{(t)})$
$C_i^{(t)} \leftarrow \mathsf{Enc}(pp, \mathbf{x}_i^{(t)}, n_i^{(t)}, t)$
return $C_i^{(t)}$

1) Establishment: All the parties agree on the public parameters $pp = (N_1, N_2, \mathcal{H}_1, \mathcal{H}_2, \mathcal{G}, T)$ where: N_1 is the modulus of encryption layer 1, N_2 is the modulus of encryption layer 2 and $N_2 > \sqrt{k} \cdot 2^{l_1}$ where l_1 is the bit-length of N_1 and k is the number of local parties; $\mathcal{H}_1 : \mathbb{Z} \to \mathbb{Z}_{N_1^2}^*, \mathcal{H}_2 : \mathbb{Z} \to \mathbb{Z}_{N_2^2}^*$ are two hash functions, $\mathcal{G}$ is the learning network and T is the number of federated learning iterations. The sub-protocol π_0 generates the secrets $\mathbf{v}^{(0)}$ as follows:

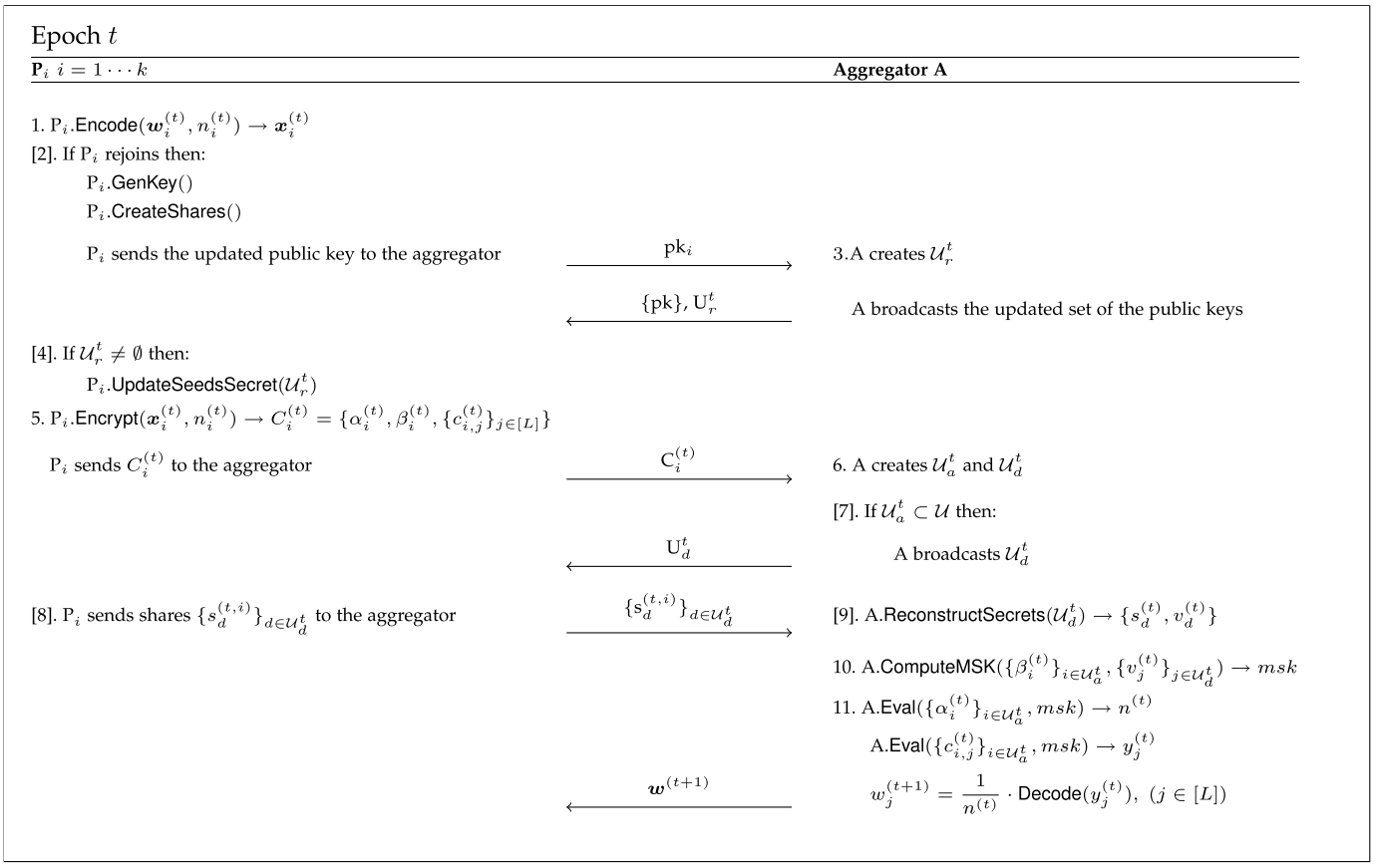

Fig. 1. Secure weighted model aggregation procedure for one epoch.

a) Sub-protocol π_0:

1. The aggregator chooses and publishes a λ-bit prime, p, where λ is the security parameter, and g is the generator of $\mathbb{Z}_p^*$
2. Each P_i uniformly samples $s_i^{(0)} \leftarrow_{\$} \mathbb{Z}_p^*$ and sends $pk_i^{(0)} = g^{s_i^{(0)}}$ to the aggregator who then broadcasts the set of all public keys to all parties
3. Each pair of clients $(\mathrm{P}_i, \mathrm{P}_j)$ computes a common pairwise seed $\gamma_{i,j}^{(0)}$:
$$\gamma_{j,i}^{(0)} = (pk_i^{(0)})^{s_j^{(0)}} = (pk_j^{(0)})^{s_i^{(0)}} = \gamma_{i,j}^{(0)} \tag{1}$$
4. Each P_i computes:
$$v_i^{(0)} = \sum_{i<j} \gamma_{i,j}^{(0)} - \sum_{i>j} \gamma_{i,j}^{(0)} \tag{2}$$
5. Each P_i runs the Shamir-secret sharing algorithm $\mathsf{SS}(s_i^{(0)}, \tau, k)$ to create k shares of its secret key $s_i^{(0)}$ and sends each triple $(i, j, s_i^{(0j)})$ to each other party P_j, where $s_i^{(0j)}$ is the share of $s_i^{(0)}$ corresponding to the party P_j:
$$\{(i, j, s_i^{(0,j)})\}_{j\in[k]} \leftarrow \mathsf{SS}(s_i^{(0)}, \tau, k) \tag{3}$$

2) Secure Weighted Aggregation: This section describes the proposed secure weighted aggregation happening at each federated learning iteration to evaluate the global model as the weighted aggregation of the encrypted local models. Fig. 1 illustrates the main steps and computations carried out during each training epoch, where a step in square brackets (e.g. [2]) indicates that this step is included if dropout/rejoining happens.

At each iteration t, each P_i owns a L-length local vector model $\boldsymbol{w}_i^{(t)}$. The following describes in detail the steps of secure weighted aggregation at the iteration $t \in [T]$

1. $\mathbf{P}_i$.Encode$(\boldsymbol{w}_i^{(t)}, n_i^{(t)}) \rightarrow \boldsymbol{x}_i^{(t)}$: P_i encodes the weighted model to get the non-negative integer vector $\boldsymbol{x}_i^{(t)}$ according to the method in [37]:
$$\boldsymbol{z}_i^{(t)} = \{n_i^{(t)} \cdot w_{i,j}^{(t)} \mid j \in [L]\}\ ;\ n_i^{(t)} = |\mathcal{D}_i^{(t)}| \tag{4}$$
$$\boldsymbol{x}_i^{(t)} = \mathsf{Encode}(\boldsymbol{z}_i^{(t)}) \tag{5}$$
$$\boldsymbol{z}_i^{(t)} = \mathsf{Decode}(\boldsymbol{x}_i^{(t)}) \tag{6}$$

[2]. If P_i rejoins this iteration, this party runs P_i.GenKey() to generate a new pair of its secret and public key, and P_i.CreateShares() to create k shares of the updated secret $s_i^{(t)}$:
$$s_i^{(t)} \leftarrow_{\$} \mathbb{Z}_p^* \tag{7}$$
$$pk_i^{(t)} = g^{s_i^{(t)}} \tag{8}$$
$$\{(i, j, s_i^{(t,j)})\}_{j\in[k]} \leftarrow \mathsf{SS}(s_i^{(t)}, \tau, k) \tag{9}$$
Then P_i sends the updated public key $pk_i^{(t)}$ to the aggregator.

3. Based on the receiving updated public keys, the aggregator creates the set of rejoining parties of this iteration, which is $\mathcal{U}_r^t$. If $\mathcal{U}_r^t = \emptyset$ then $\boldsymbol{v}^{(t)} = \boldsymbol{v}^{(t-1)}$, else the

aggregator broadcasts the updated set of the public keys $\{pk\}$ and $\mathcal{U}_r^t$.

[4]. Upon receiving $\mathcal{U}_r^t$ and $\{pk\}$, a rejoining party checks if its updated public key is in the set and then continues the protocol; if not then leaves the protocol (early dropout). If $\mathcal{U}_r^t \neq \emptyset$ then P_i.UpdateSeedsSecret().

If P_i rejoins, then P_i updates the seeds $\gamma_{i,j}^{(t)}$ shared with all other parties

$$\gamma_{i,j}^{(t)} = (pk_j^{(t)})^{s_i^{(t)}}; \quad j \in \mathcal{U} \setminus \{i\} \tag{10}$$

else P_i updates the seeds $\gamma_{i,r}^{(t)}$ shared with rejoining parties

$$\gamma_{i,r}^{(t)} = (pk_r^{(t)})^{s_i^{(t)}}; \quad j \in \mathcal{U}_r^t \setminus \{i\} \tag{11}$$

Then P_i updates their secret $v_i^{(t)}$

$$v_i^{(t)} = \sum_{i<j} \gamma_{i,j}^{(t)} - \sum_{i>j} \gamma_{i,j}^{(t)} \tag{12}$$

5. P_i.Encrypt$(\boldsymbol{x}_i^{(t)}, n_i^{(t)}) \rightarrow C_i^{(t)} = \{\alpha_i^{(t)}, \beta_i^{(t)}, \{c_{i,j}^{(t)}\}_{j\in[L]}\}$: P_i encrypts $\boldsymbol{x}_i^{(t)}$, which includes the following main steps:
   - Sample $sk_i^{(t)} \leftarrow_{\$} \pm\{0,1\}^{2l_1}$
   - Compute

$$c_{i,j}^{(t)} = (1+N_1)^{x_{i,j}^{(t)}} \cdot \mathcal{H}_1(j)^{sk_i^{(t)}} \bmod N_1^2 \quad \text{where } j = 0 \cdots L-1 \tag{13}$$

$$\alpha_i^{(t)} = (1+N_1)^{n_i^{(t)}} \cdot \mathcal{H}_1(L)^{sk_i^{(t)}} \bmod N_1^2 \tag{14}$$

$$\beta_i^{(t)} = (1+N_2)^{sk_i^{(t)}} \cdot \mathcal{H}_2(t)^{v_i^{(t)}} \bmod N_2^2 \tag{15}$$

   - Return $C_i^{(t)} = \{\alpha_i^{(t)}, \beta_i^{(t)}, \boldsymbol{c}_i^{(t)} = \{c_{i,j}^{(t)}\}_{j\in[L]}\}$

   Then P_i sends $C_i^{(t)}$ to the aggregator

6. Receiving $C_i^{(t)}$ from the alive parties, the aggregator creates the set $\mathcal{U}_a^t$ of the alive parties and $\mathcal{U}_d^t = \mathcal{U} \setminus \mathcal{U}_a^t$ of the dropped parties

[7]. If $\mathcal{U}_a^t \subset \mathcal{U}$ then the aggregator broadcasts $\mathcal{U}_d^t$

[8]. P_i sends to the aggregator the value $s_d^{(t,i)}$ which is the share of the secret $s_d^{(t)}$ of a dropped party P_d in the set $\mathcal{U}_d^t$.

[9]. A.ReconstructSecrets$(\mathcal{U}_d^t) \rightarrow \{s_d^{(t)}, v_d^{(t)}\}$: Having the Shamir's secret shares from the alive parties, the aggregator reconstructs the secret keys $s_d^{(t)}$ of dropped parties and then computes the secret $v_d^{(t)}$ of every dropped party in the set $\mathcal{U}_d^t$ from the recovered secrets.

$$v_d^{(t)} = \sum_{d<i} \gamma_{d,i}^{(t)} - \sum_{d>i} \gamma_{d,i}^{(t)}, \ d \in \mathcal{U}_d^t \quad \text{where } \gamma_{d,i}^{(t)} = pk_i^{s_d^{(t)}} \tag{16}$$

10. A. ComputeMSK$(\{\beta_i^{(t)}\}_{i\in\mathcal{U}_a^t}, \{v_j^{(t)}\}_{j\in\mathcal{U}_d^t}) \rightarrow msk$: The aggregator computes the master key $msk^{(t)}$:
   - If $\mathcal{U}_a^t = \mathcal{U}$:

$$msk^{(t)} = \frac{\prod_{i\in\mathcal{U}} \beta_i^{(t)} - 1 \bmod N_2^2}{N_2} \tag{17}$$

   - If $\mathcal{U}_a^t \subset \mathcal{U}$:

$$msk^{(t)} = \frac{(\prod_{i\in\mathcal{U}_a^t} \beta_i^{(t)}) \cdot \mathcal{H}_2(t)^{\sum_{i\in\mathcal{U}_d^t} v_i^{(t)}} - 1 \bmod N_2^2}{N_2} \tag{18}$$

11. A.Eval$(\{\alpha_i^{(t)}\}_{i\in\mathcal{U}_a^t}, msk) \rightarrow n^{(t)}$, A.Eval$(\{c_{i,j}^{(t)}\}_{i\in\mathcal{U}_a^t}, msk) \rightarrow y_j^{(t)}$
Having $msk^{(t)}$, the aggregator can compute the global model:

$$n^{(t)} = \frac{(\prod_{i\in\mathcal{U}_a^t} \alpha_i^{(t)}) \cdot \mathcal{H}_1(L)^{-msk^{(t)}} - 1 \bmod N_1^2}{N_1} \tag{19}$$

$$y_j^{(t+1)} = \frac{(\prod_{i\in\mathcal{U}_a^t} c_{i,j}^{(t)}) \cdot \mathcal{H}_1(j)^{-msk^{(t)}} - 1 \bmod N_1^2}{N_1} \tag{20}$$

$$w_j^{(t+1)} = \frac{1}{n^{(t)}} \cdot \mathsf{Decode}(y_j^{(t)}), \ (j \in [L]) \tag{21}$$

Then, the aggregator sends the global model $\boldsymbol{w}^{(t+1)} = \{w_j^{(t+1)}\}_{j\in[L]}$ to all local parties for the next epoch $t+1$.

V. Analysis of the Proposed Scheme

A. Correctness

- If $\mathcal{U}_a^t = \mathcal{U}$:
From (25):

$$\prod_{i\in\mathcal{U}_a^t} \beta_i^{(t)} = \prod_{i\in\mathcal{U}_a^t} (1+N_2)^{sk_i^{(t)}} \cdot \mathcal{H}_2(t)^{v_i^{(t)}} \bmod N_2^2 = (1+N_2)^{\sum_{i\in\mathcal{U}_a^t} sk_i^{(t)}} \cdot \mathcal{H}_2(t)^{\sum_{i\in\mathcal{U}_a^t} v_i^{(t)}} \bmod N_2^2 \tag{22}$$

Besides, from (2, 12), we have:

$$\sum_{i\in\mathcal{U}} v_i^{(t)} = 0 \tag{23}$$

Thus,

$$\prod_{i\in\mathcal{U}_a^t} \beta_i^{(t)} = (1+N_2)^{\sum_{i\in\mathcal{U}_a^t} sk_i^{(t)}} \bmod N_2^2 \tag{24}$$

From (17):

$$\begin{aligned} msk^{(t)} &= \frac{\prod_{i\in\mathcal{U}_a^t} \beta_i^{(t)} - 1 \bmod N_2^2}{N_2} \\ &= \frac{(1+N_2)^{\sum_{i\in\mathcal{U}_a^t} sk_i^{(t)}} \bmod N_2^2 - 1 \bmod N_2^2}{N_2} \\ &= \frac{(1+\sum_{i\in\mathcal{U}_a^t} sk_i^{(t)} \cdot N_2) \bmod N_2^2 - 1 \bmod N_2^2}{N_2} \\ &= \sum_{i\in\mathcal{U}_a^t} sk_i^{(t)} \bmod N_2^2 \end{aligned} \tag{25}$$

- If $\mathcal{U}_a^t \subset \mathcal{U}$
Based on the Shamir threshold secret sharing scheme, the aggregator can reconstruct all the secrets $s_d^{(t)}$ of dropped

parties as long as the aggregator receives at least τ shares of each $s_d^{(t)}$. From that, the aggregator can recover $v_i^{(t)}$ of dropped parties and obtains $\sum_{i\in\mathcal{U}_d^t} v_i^{(t)}$.

$$\prod_{i\in\mathcal{U}_a^t} \beta_i^{(t)} = (1+N_2)^{\sum_{i\in\mathcal{U}_a^t} sk_i^{(t)}} \cdot \mathcal{H}_2(t)^{\sum_{i\in\mathcal{U}_a^t} v_i^{(t)}} \bmod N_2^2 \tag{26}$$

Substitute (26) into (18), and note that $\sum_{i\in\mathcal{U}_a^t} v_i^{(t)} + \sum_{i\in\mathcal{U}_d^t} v_i^{(t)} = \sum_{i\in\mathcal{U}} v_i^{(t)} = 0$, we have:

$$\begin{aligned} msk^{(t)} &= \frac{(\prod_{i\in\mathcal{U}_a^t} \beta_i^{(t)}) \cdot \mathcal{H}_2(t)^{\sum_{i\in\mathcal{U}_d^t} v_i^{(t)}} - 1 \bmod N_2^2}{N_2} \\ &= \frac{(1+N_2)^{\sum_{i\in\mathcal{U}_a^t} sk_i^{(t)}} \cdot \mathcal{H}_2(t)^0 - 1 \bmod N_2^2}{N_2} \\ &= \frac{(1+\sum_{i\in\mathcal{U}_a^t} sk_i^{(t)} \cdot N_2) \bmod N_2^2 - 1 \bmod N_2^2}{N_2} \\ &= \sum_{i\in\mathcal{U}_a^t} sk_i^{(t)} \bmod N_2^2 \end{aligned} \tag{27}$$

Hence, in both cases, we successfully compute the master key $msk^{(t)} = \sum_{i\in\mathcal{U}_a^t} sk_i^{(t)} \bmod N_2^2$

From $N_2 > \sqrt{k} \cdot 2^{l_1}$ and $sk_i^{(t)} < 2^{2l_1}$, we have:

$$N_2^2 > k \cdot 2^{2l_1} > \sum_{i\in\mathcal{U}_a^t} sk_i^{(t)}$$

Then

$$msk^{(t)} = \sum_{i\in\mathcal{U}_a^t} sk_i^{(t)} \tag{28}$$

Next, we prove that with this master key, the global model can be correctly computed. In fact, from (13, 14) we have:

$$\prod_{i\in\mathcal{U}_a^t} \alpha_i^{(t)} = (1+N_1)^{\sum_{i\in\mathcal{U}_a^t} n_i^{(t)}} \cdot \mathcal{H}_1(L)^{\sum_{i\in\mathcal{U}_a^t} sk_i^{(t)}} \bmod N_1^2 \tag{29}$$

$$\prod_{i\in\mathcal{U}_a^t} c_{i,j}^{(t)} = (1+N_1)^{\sum_{i\in\mathcal{U}_a^t} x_{i,j}^{(t)}} \cdot \mathcal{H}_1(j)^{\sum_{i\in\mathcal{U}_a^t} sk_i^{(t)}} \bmod N_1^2 \tag{30}$$

Substitute (28, 29) into (19), we have:

$$\begin{aligned} n^{(t)} &= \frac{(\prod_{i\in\mathcal{U}_a^t} \alpha_i^{(t)}) \cdot \mathcal{H}_1(L)^{-msk^{(t)}} - 1 \bmod N_1^2}{N_1} \\ &= \frac{(1+\sum_{i\in\mathcal{U}_a^t} n_i^{(t)} \cdot N_1) \bmod N_1^2 - 1 \bmod N_1^2}{N_1} \\ &= \sum_{i\in\mathcal{U}_a^t} n_i^{(t)} \bmod N_1^2 \end{aligned} \tag{31}$$

Similarly, substitute (28, 30) into (20), we have:

$$\begin{aligned} y_j^{(t+1)} &= \frac{(\prod_{i\in\mathcal{U}_a^t} c_{i,j}^{(t)}) \cdot \mathcal{H}_1(j)^{-msk^{(t)}} - 1 \bmod N_1^2}{N_1} \\ &= \frac{(1+\sum_{i\in\mathcal{U}_a^t} x_{i,j}^{(t)} \cdot N_1) \bmod N_1^2 - 1 \bmod N_1^2}{N_1} \\ &= \sum_{i\in\mathcal{U}_a^t} x_{i,j}^{(t)} \bmod N_1^2 \end{aligned} \tag{32}$$

$$\begin{aligned} w_j^{(t+1)} &= \frac{1}{n^{(t)}} \cdot \mathsf{Decode}(y_j^{(t+1)}) \\ &= \frac{1}{n^{(t)}} \cdot \mathsf{Decode}(\sum_{i\in\mathcal{U}_a^t} x_{i,j}{}^{(t)}) \\ &= \frac{1}{n^{(t)}} \cdot \sum_{i\in\mathcal{U}_a^t} z_{i,j}{}^{(t)} \text{(from (6))} \\ &= \frac{1}{n^{(t)}} \cdot \sum_{i\in\mathcal{U}_a^t} n_i^{(t)} \cdot w_{i,j}^{(t)} \text{ (from (4))} \end{aligned} \tag{33}$$

This proves that the aggregator can compute the global model as the weighted average of all local models even if the aggregator does not know the true value of each local model.

B. Security Analysis

In this section, we prove that the proposed protocol is secure multiparty computation against an honest-but-curious adversary who controls the aggregator server and a set $\mathcal{C}$ of colluded parties where $|\mathcal{C}| < \tau$. The aggregator is always online while participants P_i may drop out and rejoin at any iteration.

The security guarantee of the proposed scheme is based on Shamir's secret sharing scheme, and the aggregator obliviousness security provided by the encryption construction in [19] under DCR assumption in the random oracle model. Security is against a computationally-bounded honest-but-curious aggregator server.

We will consider the executions of the proposed protocol where an honest-but-curious aggregator server interacts with a set of parties, the underlying encryption construction is based on DCR assumption, and the Shamir secret sharing's threshold is set to τ. In such executions, users might drop and rejoin at any iteration. The following proves the indistinguishability of the distribution of the random variable representing the adversary view in a real execution of the proposed protocol and the distribution of the random variable representing the adversary view in a secure-by-definition "ideal world" using a *simulation-based proof*, which is a standard for security analysis of multiparty computation protocol [38]. The security analysis of the protocol indicates that what the adversary learns from the real protocol execution is no more than what she can learn from the ideal protocol execution which provides security/privacy. This also means the protocol in real execution is secure against an honest-but-curious adversarial model. To be more specific, the joint view of the server and any set of less than τ clients does not leak any information about the other clients' inputs (i.e., locally trained models/local

training data) besides what can be inferred from the output of the protocol computation (i.e., the aggregate model).

Let $\mathsf{REAL}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}$ be a random variable representing the view of the adversary in a real execution of the proposed protocol. Let $\mathcal{S}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}$ be the view of the adversary generated by a simulator in a secure-by-definition "ideal world". It is going to be proved that the distributions of $\mathsf{REAL}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}$ and $\mathcal{S}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}$ are indistinguishable.

$$\{\mathsf{REAL}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}\} \cong \{\mathcal{S}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}\}$$

We use the hybrid argument technique to prove this. First, we define a series of hybrid random variables $\mathsf{H}_0, \mathsf{H}_1, \cdots$ to construct the simulator $\mathcal{S}$ in an "ideal world" by the subsequent modifications such that any two subsequent random variables H_i and H_{i+1} are computationally indistinguishable, starting from H_0 which is the same as $\mathsf{REAL}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}$. The final result of subsequent modification is $\mathcal{S}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}$.

- H_0: This random variable is distributed exactly as $\{\mathsf{REAL}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}\} \cong \{\mathsf{H}_0\}$
- H_1: This hybrid is distributed exactly as H_0, but shares of 0 (using a different sharing of 0 for every honest party) substitute for all shares of $s_i^{(t)}$ generated by honest parties and given to the corrupted parties. Since the adversaries in $\mathcal{C}\cap\mathcal{A}$ do not receive any additional shares of $s_i^{(t)}$ from an honest party, the combined view of adversaries has only $|\mathcal{C}| < \tau$ shares of each secret $s_i^{(t)}$. The security properties of Shamir's secret sharing guarantee that the distribution of any shares of 0 is identical to the distribution of an equivalent number of shares of any given secret $s_i^{(t)}$, making this hybrid identically distributed to H_0, $\{\mathsf{H}_0\} \cong \{\mathsf{H}_1\}$
- H_2: In this hybrid, compared to H_1, for each honest party P_i, the ciphertexts $c_{i,j}^{(t)}, t \in [T]$ of $x_{i,j}^{(t)}$ is replaced by the cipher text of a dummy vector $\mathbf{0}$, the ciphertexts $\alpha_i^{(t)}, t \in [T]$ of $n_i^{(t)}$ is replaced by the ciphertext of a dummy value 0; hash function $\mathcal{H}_1$ is substituted with a truly random function $\mathcal{O}_1$. The aggregator obliviousness security in the random-oracle model under the DCR assumption of the construction in [19] guarantees that this hybrid is indistinguishable from the previous one, $\{\mathsf{H}_1\} \cong \{\mathsf{H}_2\}$
- H_3: In this hybrid, compared to H_2, for each honest party, $v_i^{(t)}$ is replaced by random $y_i^{(t)}$ subject to $\sum_{i\in\mathcal{U}\setminus\mathcal{C}} y_i^{(t)} = -\sum_{j\in\mathcal{C}} v_j^{(t)}$; and hash function $\mathcal{H}_2$ is substituted with a truly random function $\mathcal{O}_2$. The aggregator obliviousness security in the random-oracle model under the DCR assumption of the construction in [19] guarantees that this hybrid is indistinguishable from the previous one, $\{\mathsf{H}_2\} \cong \{\mathsf{H}_3\}$

Defining such a simulator $\mathcal{S}$ as described in the last hybrid, the view generated by $\mathcal{S}$ is computationally indistinguishable from that of the real execution: $\{\mathsf{REAL}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}\} \cong \{\mathsf{H}_0\} \cong \{\mathsf{H}_1\} \cong \{\mathsf{H}_2\} \cong \{\mathsf{H}_3\} \cong \{\mathcal{S}_{\mathcal{C}\cap\mathcal{A}}^{\mathcal{U},\tau,\lambda}\}$.

C. Communication and Computation Analysis

Communication and computation overheads are analyzed according to the establishment phase and each iteration of federated learning where there is $k_r (0 \le k_r < k)$ rejoined parties (with or without P_i) and $k_d (0 \le k_d < k)$ dropped parties. The computation and communication overheads are summarized in Table IV and Table V, respectively. Denote $l_{pk}, l_{ss}, l_i, l_{e1}, l_{e2}, l_p$ are the sizes in bits of a public key, a secret share, an integer, a first-layered ciphertext, a second-layered ciphertext, and a plaintext, respectively. The cost in the square brackets ([]) is included in the case of dropouts/rejoins happens.

TABLE IV
COMPUTATION OVERHEAD OF EACH LOCAL PARTY AND THE AGGREGATOR AT THE ESTABLISHMENT PHASE AND EACH ITERATION. THE EXPRESSIONS IN [] ARE INCLUDED IN THE CASE OF DROPOUT/REJOINING HAPPENS

	Establishment	Each iteration
Local party	$O(\tau \cdot k)$	$O(L + [\tau \cdot k])$
Aggregator	n/a	$O(L + [k^2])$

1) Computation Cost:

a) Computation cost of a local party: The computation cost of each party P_i at the establishment phase includes the main parts: 1- generating its public key, 2- performing each pair-wise secret agreement with each of other $k-1$ parties, which takes $O(k-1)$, and 3- creating τ-out-of-k Shamir secret shares of $s_i^{(t)}$ which is $O(\tau \cdot k)$. Thus, the computation cost of each party P_i at the establishment phase is $O(\tau \cdot k)$.

P_i's computation cost at each iteration is the cost of creating the ciphertexts $c_{i,j}^{(t)}$, $\alpha_i^{(t)}, \beta_i^{(t)}$ which takes $O(L)$. If P_i rejoins, then there is extra computation cost as the cost of P_i in the establishment phase, which is $O(\tau \cdot k)$. Thus the total computation of each party in an iteration is $O(L + [\tau \cdot k])$.

b) Computation cost of the aggregator: The aggregator's computation cost can be divided into the main operations: 1- reconstructing Shamir secrets (one for each dropped party) whenever dropouts happen, which takes the total time $O(k^2)$, and 2- obtaining $\mathbf{w}^t$ by carrying decryption $O(L)$ times. Thus the total computation cost of the aggregator at an iteration is $O(L + [k^2])$.

2) Communication Cost:

a) Communication cost of a local party: The communication cost of each party P_i at the establishment phase includes the main parts: sending its public key to the aggregator, sending $k-1$ secret shares to other $k-1$ parties (each secret share to each party), resulting $l_{pk} + (k-1) \cdot l_{ss}$, which is $O(k)$.

The communication cost of each party P_i at an iteration can be partitioned into the main parts: 1- receiving k updated public keys from the aggregator, which takes $k \cdot l_{pk}$, 2- sending $k-1$ secret shares of its updated secret $s_i^{(t)}$ when it rejoins which takes $(k-1) \cdot l_{ss}$, 3- sending its secret shares of k_d dropped parties' secrets which is $k_d \cdot l_{ss}$, 4- sending an encryption message $C_i^{(t)} = \{\alpha_i^{(t)}, \beta_i^{(t)}, \boldsymbol{c}_i^{(t)} = \{c_{i,j}^{(t)}\}_{j\in[L]}\}$ to the aggregator at every iteration t, which accounts for $(l_{e1} + l_{e2} + L \cdot l_{e_1})$, and 5- receiving the aggregate model, which is $L \cdot l_p$. Thus, communication cost of P_i at an iteration includes: download cost (i.e., receiving messages) is $[k \cdot l_{pk}] + L \cdot l_p$ or

TABLE V
COMMUNICATION OVERHEAD OF EACH LOCAL PARTY AND THE AGGREGATOR AT THE ESTABLISHMENT PHASE AND EACH ITERATION. THE EXPRESSIONS IN [] IS INCLUDED IN THE CASE OF DROPOUTS/REJOINS HAPPENS

	Establishment		Each iteration	
	Download	Upload	Download	Upload
Local party	n/a	$O(k)$	$O(L+[k])$	$O(L+[k])$
Aggregator	$O(k)$	n/a	$O(L\cdot k+[k^2])$	$O(L\cdot k+[k^2])$

$O(L+[k])$, upload cost (i.e, sending messages) is $[(k-1)\cdot l_{ss}]+[k_d\cdot l_{ss}]+(l_{e1}+l_{e2}+L\cdot l_{e_1}) < [(2k-1)\cdot\lambda]+(l_{e1}+l_{e2}+L\cdot l_{e_1})$ ($l_{pk}=l_{ss}=\lambda$), or $O(L+[k])$.

b) Communication cost of the aggregator: The communication cost of the aggregator at establishment phase includes receiving k public keys of k parties, resulting $k\cdot l_{pk}$, which is $O(k)$.

The communication cost of the aggregator at an iteration can be broken into the main parts: 1- receiving k_r updated public keys which is $k_r\cdot l_{pk}$, 2- sending the updated set of public keys to k parties which is $k\cdot k\cdot l_{pk}$, 3- receiving secret shares of the dropped parties from the alive parties, which causes maximum $(k-k_d)\cdot k_d\cdot l_{ss}$, 4- receiving $k-k_d$ encryption message which is $(k-k_d)\cdot(l_{e1}+l_{e2}+L\cdot l_{e_1})$, and 5- sending the aggregate model to each local party, which is $k\cdot L\cdot l_p$. Thus, the communication cost of the aggregator at an iteration includes: upload cost is $[k^2]+k\cdot L\cdot l_p$ or $O(L\cdot k+[k^2])$, download cost is $[k_r\cdot l_{pk}]+[(k-k_d)\cdot k_d\cdot l_{ss}]+k\cdot(l_{e1}+l_{e2}+L\cdot l_{e1})-[k_d\cdot(l_{e1}+l_{e2}+L\cdot l_{e1})] < [k\cdot l_{pk}]+[k^2/4\cdot l_{ss}]+k\cdot(l_{e1}+l_{e2}+L\cdot l_{e1}) = [(\frac{k^2}{4}+k)\cdot\lambda]+k\cdot(l_{e1}+l_{e2}+L\cdot l_{e1})$, resulting in $O(L\cdot k+[k^2])$.

TABLE VI
PARAMETERS FOR LAYERS OF THE NETWORK IN FIG. 2

Layer	Kernel Size	Kernel Numbers	Batch Normalization	Activation Function
$f^{c\text{-}n\text{-}a}_{1\text{-}b}$	1×3	12	Yes	ELU
$f^{c\text{-}n\text{-}a}_{1\text{-}l}$	1×3	12	Yes	ELU
$f^{c\text{-}n\text{-}a}_{2\text{-}b}$	1×12	4	Yes	ELU
$f^{c\text{-}n\text{-}a}_{2\text{-}l}$	1×12	4	Yes	ELU
$f^{c\text{-}n\text{-}a}_{3\text{-}bl}$	$n_{bl}\times 4$	$2n_{bl}$	Yes	ELU
$f^{l\text{-}a}_{6\text{-}bl}$	$4n_{bl}\times 1$	1	no	sigmoid

TABLE VII
PARAMETERS FOR LSTM LAYERS OF THE NETWORK IN FIG. 2

Layer	Input Size	Output Size
$f^{LSTM}_{4\text{-}bl}$	$2n_{bl}$	$2n_{bl}$
$f^{LSTM}_{5\text{-}bl}$	$2n_{bl}$	$4n_{bl}$

VI. PRIVACY-ENHANCING CROSS-SILO FEDERATED LEARNING FDIA DETECTION IN SMART GRIDS

Consider a multi-area grid of k non-overlapping areas managed by k independent transmission grid companies (TGCs). There is a system operator (SO) who takes care of the interconnection areas and coordinates operations. Each P_i TGC owns a private local dataset $\mathcal{D}_i$, $i\in\{1,\ldots,k\}$ with $n_i=|\mathcal{D}_i|$ samples and has communication lines with the SO and other TGCs.

For FDIA detection in smart grids, the federated learning approach is superior to the centralised in terms of data privacy protection and communication overhead. From a data privacy protection viewpoint, the private data of each local party are not transmitted outside of federated learning, while for a centralized approach, all these data have to be uploaded to a central server, which is a risk to more security threats. From a communication overhead perspective, in federated learning, local models are transmitted to the centre instead of raw measurement data. This also helps reduce communication overhead due to the fact that the size of models is often much smaller than raw measurement data.

An honest-but-curious adversarial model is considered. Adversaries are assumed to be honest but curious in the sense that they follow the protocol but can obtain available transcripts to learn extra information that should remain private. A good result of detecting false data injection attacks supporting security operations and power management is a common interest of all parties, thus it is reasonable that they are incentivised to follow the protocol to achieve the best output. However, some parties might be motivated to conspire with each other to infer private training data samples of a target party for some business benefits. In the context of the above-proposed system model, a semi-honest adversary is an adversary that controls SO and a set of colluded TGCs.

To model the spatial-temporal relationship between bus and line measurements, a network architecture modified from the method [36] is trained for the FDIA detection, as shown in Fig. 2. The model in Fig. 2 is utilized to detect false data injection attacks in transmission power grids. In the training stage, the model is securely trained by the proposed privacy-enhancing cross-silo federated learning framework. The trained global model is then distributed to each participant/sub-grid. In the test stage, each sub-grid utilizes the trained global model to detect FDIAs individually. Time-series bus measurements $\mathbf{Z}^b_{t_i}$ and transmission line measurements $\mathbf{Z}^l_{t_i}$ are fed into the model, which is utilized to model the spatial-temporal relationship between bus and line measurements. The model will output the likelihood of FDIAs in the current sub-grid. The details of network parameters are summarised in Table VI and Table VII.

With the above training network architecture, the training network model for FDIA detection has 132743 parameters.

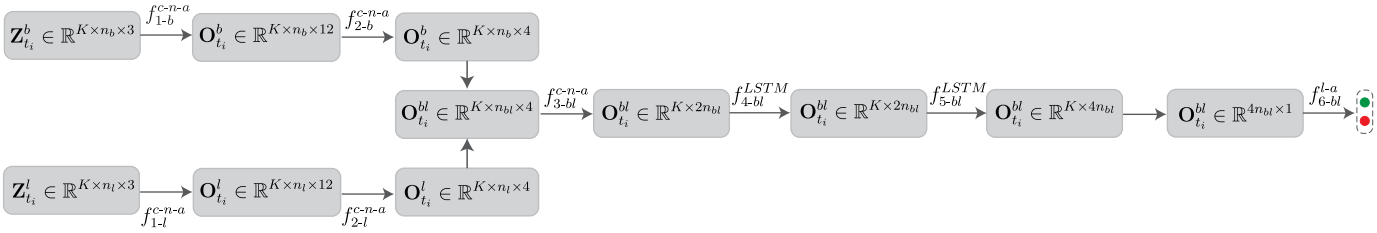

Fig. 2. The network architecture for FDIA detection in AC-model transmission power grids.

TABLE VIII
DETAILS ABOUT THE TRANSMISSION POWER GRID '1-HV-MIXED-0-NO SW'

Component	Quantity	Explanation
Bus	64	all the buses are in service.
Load	58	
Static Generator	103	
Lines	95	all the lines are in service.
Transformer	6	
External grid	3	supply/generate power
demand profiles	35,136	power profiles for one year
Measurements	355	bus active/reactive power injection, bus voltage magnitude, line active/reactive power flows, and line electrical current

The proposed privacy-enhancing cross-silo FDIA detection is based on the classical federated learning framework FedAvg [1] with the privacy protection part on top.

VII. EMPIRICAL EVALUATION

This section demonstrates the desirable utility and efficiency of the proposed cross-silo privacy-enhancing federated learning. In the following, we provide the description of the measurement dataset and the transmission power grid system which includes several subgrids controlled by local TGCs and a SO who coordinates the federated learning process. Following that is the training/testing setting and the discussion of the performance in terms of accuracy, training time and inference time.

A. Description of Datasets

1) Transmission Power Grid Test Set: A transmission power grid, '1-HV-mixed-0-no sw', from the benchmark dataset SimBench [39] was used to evaluate the FDIA detection. This power grid contains 64 buses, 58 loads, and 355 measurements, with more details shown in Table VIII. This power grid is divided into four sub-grids, with each sub-grid containing 16 buses, summarised as follows:

- sub-grid S_1 contains bus 62, 26, 130, 44, 94, 104, 74, 48, 106, 50, 12, 54, 52, 0, 56, 34,
- sub-grid S_2 contains bus 64, 100, 128, 68, 110, 112, 126, 66, 122, 98, 124, 102, 38, 96, 116, 120,
- sub-grid S_3 contains bus 92, 22, 84, 14, 20, 76, 132, 18, 114, 4, 80, 90, 42, 40, 82, 28, and
- sub-grid S_4 contains bus 32, 2, 36, 70, 72, 108, 46, 78, 16, 86, 118, 24, 58, 88, 60, 30.

For each sub-grid, the bus measurements consist of active/reactive power injections and bus voltage magnitude, denoted by $\boldsymbol{z}^b_{t_k} \in \mathbb{R}^{n_b\times 3}$ at time t_k; and the line measurements consist of active/reactive power flows and line electrical current, denoted by $\boldsymbol{z}^l_{t_k} \in \mathbb{R}^{n_l\times 3}$ at time t_k.

2) Normal and FDIA Measurement Data: The power grid '1-HV-mixed-0-no sw' contains 35136 demand profiles, with one profile per 15 minutes for one year. To generate the datasets which include the normal measurement and the FDIA measurement, the commercial software PowerFactory 2017 SP4,[1] the open source software Pandapower,[2] and the benchmark SimBench [3] were utilised. The normal measurements were obtained by calculating the power flow using the commercial software PowerFactory 2017 SP4. The attacks were launched on a target bus by modifying either its voltage angle or voltage magnitude. All of these FDIA measurement samples have bypassed the residual-based data detection function of PowerFactory 2017 SP4.

B. Training and Testing Setting

There are 35136 normal measurement samples and 35136 FDIA measurement samples, with normal measurement samples labelled 0 and FDIA samples labelled 1. In the training stage, 29952 normal samples and 29952 FDIA samples for the first 312 days are grouped as the training dataset; the other 5184 normal and FDIA samples for the remaining 54 days are used as the test dataset. In the federated learning training, the number of global epochs was set to 200, the number of local epochs was set to 5, the number of local batches was set to 48, and the sequence number for LSTM layers is set to 96. In each federated learning training round, 3 local sub-grids were randomly selected to collaboratively train the global model. The federated learning source code [4] and the popular deep learning framework Pytorch-1.9.0[5] were used to implement the proposed FDIA federated learning detection framework for the model training and testing.

Three commonly used metrics were applied to evaluate the accuracy of the FDIA detection, namely precision, recall, and

[1] https://www.digsilent.de/en/powerfactory.html
[2] https://www.pandapower.org/
[3] https://simbench.readthedocs.io/en/stable/about/installation.html
[4] https://github.com/AshwinRJ/Federated-Learning-PyTorch
[5] https://pytorch.org/docs/1.9.0/

TABLE IX
CENTRALIZED TRAINED FDIA DETECTION ACCURACY

Precision (%)	*Recall* (%)	F_1 (%)
98.515	97.261	97.884

TABLE X
FEDAVG FDIA DETECTION ACCURACY

Sub-grid	*Precision* (%)	*Recall* (%)	F_1 (%)
S_1	97.472	96.701	97.085
S_2	98.167	96.103	97.124
S_3	97.865	96.393	97.123
S_4	97.098	96.798	96.947

F_1 score, expressed by

$$\begin{cases} Precision = \dfrac{N_{tp}}{N_{tp} + N_{fp}}, \\ Recall = \dfrac{N_{tp}}{N_{tp} + N_{fn}}, \\ F_1 = 2 \times \dfrac{Precision \times Recall}{Precision + Recall}, \end{cases}$$

where N_{fp} indicates the number of false positive, N_{tp} indicates the number of true positive, N_{fn} indicates the number of false negative, and N_{tn} indicates the number of true negative.

C. FDIA Detection Accuracy and Time Overhead

We have compared the performance of the proposed solution (i.e. the federated learning trained model on encrypted local models from each local dataset) with the centralized trained model on the whole plain dataset. The same model was trained, without the proposed encryption scheme, in the centralized way using the same hyperparameters in Section VII-B. The results of the centralized trained model on the whole plain dataset are summarized in Table IX. Table X is for the FDIA detection accuracy of FedAvg FDIA detection algorithm on the test dataset. As can be seen from Table IX and Table X, there is no big difference in the accuracy.

The privacy-enhancing FedAvg FDIA detection version has the same accuracy as the original FedAvg FDIA detection version. However, the average training time for each sub-grid as well as for the whole system to get the weighted global model is longer due to the complexity of privacy protection added for secure weighted aggregation. The average training time is collected by evaluating the framework in a Linux system with each sub-grid using one Nvidia Tesla Volta V100-SXM2-32GB GPU.

Encryption parameters are set as: $\lambda = 2048$ (modulus p in the sub-protocol π_0 is a 2048-bit prime), $l_1 = 256$ (modulus N_1 of the first encryption layer is 256-bit length integer), $l_2 = 512$ (modulus N_2 of the second encryption layer is a 512-bit length), $l_p = 64$.

For each federated learning round, each TGC timed its own part including the local model training part and the privacy protection part; SO timed the section of obtaining the encrypted aggregation model and decrypting it. In Table XI we provide the average computational time in seconds per one global epoch (one federated learning round) of our proposed privacy-enhancing FDIA detection federated learning in a single-processing manner. The local model training part without privacy protection consumes around 233 seconds. The average extra time for the privacy protection part comprises 1- the time for the initial setting of the protection scheme which is 16.41 seconds on average, 2- the computation time of local model protection which happens at the client side at every federated learning round which is 12.35 seconds in average per client per round, 3- the computation time of obtaining the encrypted aggregation model and decrypting it which happens at the server side at every federated learning round which is 12.14 seconds in average per round.

TABLE XI
AVERAGE COMPUTATIONAL TIME IN SECONDS PER ONE GLOBAL EPOCH IN A SINGLE-PROCESSING MANNER

Avg. TGC's time for model protection	**Avg. SO's time for model aggregation and decryption**
$\sim$ 12.35 seconds	$\sim$ 12.14 seconds

TABLE XII
AVERAGE COMPUTATIONAL TIME IN SECONDS PER ONE GLOBAL EPOCH IN A MULTI-PROCESSING MANNER WITH 4 CPUS

Avg. TGC's time for model protection	**Avg. SO's time for model aggregation and decryption**
$\sim$ 5.56 seconds	$\sim$ 5.24 seconds

To test the ability to accelerate the computation time, the multiprocessing technique is implemented to partition the Singular Instruction Multiple Data (SIMD) computations of cryptography operations over model vectors onto 4 CPUs. Table XII illustrates the possibility of accelerating the speed by multiprocessing utilizing 4 CPUs. The computation overhead of local model protection in each federated learning round with security on top only incurs 5.56 seconds, i.e., 2.38% compared to 233 seconds of the underlying model without security. The total extra time of the privacy protection component running over 200 epochs of federated learning training in a single-processing manner is around 83 minutes, while in a multi-processing manner with 4 CPUs is around 36 minutes. The implementation of our proposed scheme is well-suited for parallel computation. Thus, the extra computational time overhead that occurred from our privacy-protection component could be significantly reduced by using more CPUs that local transmission grid operators are facilitated or from the cloud at the very low price.[6]

From the communication analysis in Section V-C.2, with the above encryption parameter setting for the experiment and the size of model vector is $L = 132743$, the download cost of a client is less than $k \cdot \lambda + L \cdot l_p = 4 \cdot 2048 + 132743 \cdot 64 = 8503744$ bits ≈ 8.5 Mbits $= 1$ Mbyte, the upload cost of

[6]https://aws.amazon.com/ec2/pricing/on-demand/

a client is less than $[(2k-1)\cdot\lambda]+(l_{e_1}+l_{e_2}+L\cdot l_{e_1}) = (2\cdot 4-1)\cdot 2048+(512+1024+132743\cdot 512)\approx 68$ Mbits $=$ 8.5 Mbytes;

The model training is not a real-time process, thus we can afford more time for transmission leading to a lower bandwidth. If 1 second per iteration is used for uploading data from a local party to the aggregator (resulting in 0.05 hours of uploading data from a local party to the aggregator in the whole training process with 200 epochs used in the experiment), then the upload bandwidth requirement would be 68Mbps. The network bandwidth for our campus office is 900Mbps.

In the inference stage, each sub-grid utilizes the trained global model to detect FDIAs individually. Time-series bus measurements $\mathbf{Z}^b_{t_i}$ and transmission line measurements $\mathbf{Z}^l_{t_i}$ are fed into the model, which is utilized to model the spatial-temporal relationship between bus and line measurements. The model will output the likelihood of FDIAs in the current sub-grid. Detecting FDIA given a trained model (i.e., inference) in the proposed scheme is 6.7 milliseconds on average, which is fast for relevant smart grid operations, e.g., state estimation.

VIII. Conclusion

In this paper, we propose a cross-silo privacy-enhancing federated learning which is secure in the honest-but-curious adversarial model. With the main techniques of secure multi-party computation based on double-layer encryption and secret sharing, the scheme is efficient in communication and computation overhead and robust against dropouts and rejoining. The scheme removes the requirement of computing discrete logarithms or multiple non-colluding server settings which are the limitations of some related works. In addition, the client's secret keys of two encryption layers are generated by each party in a decentralized manner which helps increase the level of privacy guarantee. We also firstly design and empirically evaluate a practical and efficient privacy-enhancing cross-silo federated learning resilient to the local private data inference attacks for FDIA detection in the smart grid domain. The proposed scheme provides a framework which can be adapted to other domains. The analysis of security and the empirical evaluation proves that the proposed scheme achieves provable privacy against an honest-but-curious aggregator server colluding with some clients while providing desirable model utility in an efficient manner. In future works, we are going to investigate more different adversarial models in various federated learning settings which is applicable for security in cyber-physical systems.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statistics. (AISTATS)*, Fort Lauderdale, FL, USA, in Proceedings of Machine Learning Research, vol. 54, A. Singh and J. Zhu, Eds. PMLR, Apr. 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a?ref=https://githubhelp.com.

[2] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1322–1333.

[3] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 601–618.

[4] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 603–618.

[5] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, Dec. 2019, pp. 148–162.

[6] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 691–706.

[7] N. Carlini, C. Liu, U. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 267–284.

[8] G. Hug and J. A. Giampapa, "Vulnerability assessment of AC state estimation with respect to false data injection cyber-attacks," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1362–1370, Sep. 2012.

[9] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong, "A review of false data injection attacks against modern power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1630–1638, Jul. 2017.

[10] R. D. Christie, B. F. Wollenberg, and I. Wangensteen, "Transmission management in the deregulated environment," *Proc. IEEE*, vol. 88, no. 2, pp. 170–195, Feb. 2000.

[11] F. Karmel, "Deregulation and reform of the electricity industry in Australia," Aust. Government-Dept. Foreign Affairs Trade, Barton, ACT, Australia, Aust.-Jpn. Found. Grant 2017-18, 2018. [Online]. Available: https://www.dfat.gov.au/sites/default/files/deregulation-of-the-energy-industry-australian-experience.pdf

[12] L. Sankar, "Competitive privacy: Distributed computation with privacy guarantees," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Dec. 2013, pp. 325–328.

[13] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, Jun. 2021.

[14] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–487, 2013.

[15] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1310–1321.

[16] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.

[17] A. G. Sébert, R. Sirdey, O. Stan, and C. Gouy-Pailler, "Protecting data from all parties: Combining FHE and DP in federated learning," 2022, *arXiv:2205.04330*.

[18] E. Shi, T. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. NDSS*, vol. 2, 2011, pp. 1–17.

[19] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2013, pp. 111–125.

[20] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.

[21] J. Guo, Z. Liu, K.-Y. Lam, J. Zhao, and Y. Chen, "Privacy-enhanced federated learning with weighted aggregation," in *Proc. Int. Symp. Secur. Privacy Social Netw. Big Data*. Cham, Switzerland: Springer, 2021, pp. 93–109.

[22] H. Corrigan-Gibbs and D. Boneh, "Prio: Private, robust, and scalable computation of aggregate statistics," in *Proc. 14th USENIX Symp. Networked Syst. Design Implement.*, 2017, pp. 259–282.

[23] H. Fereidooni et al., "SAFELearn: Secure aggregation for private federated learning," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2021, pp. 56–62.

[24] Y. Dong, X. Chen, L. Shen, and D. Wang, "EaSTFLy: Efficient and secure ternary federated learning," *Comput. Secur.*, vol. 94, Jul. 2020, Art. no. 101824.

[25] C. Fang, Y. Guo, N. Wang, and A. Ju, "Highly efficient federated learning with strong privacy preservation in cloud computing," *Comput. Secur.*, vol. 96, Sep. 2020, Art. no. 101889.

[26] S. Truex et al., "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, Nov. 2019, pp. 1–11.

[27] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, Nov. 2019, pp. 13–23.

[28] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly)logarithmic overhead," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1253–1269.

[29] A. Madi, O. Stan, A. Mayoue, A. Grivet-Sebert, C. Gouy-Pailler, and R. Sirdey, "A secure federated learning framework using homomorphic encryption and verifiable computing," in *Proc. Reconciling Data Anal., Automat., Privacy, Secur., Big Data Challenge (RDAAPS)*, May 2021, pp. 1–8.

[30] W.-T. Lin, G. Chen, and Y. Huang, "Incentive edge-based federated learning for false data injection attack detection on power grid state estimation: A novel mechanism design approach," *Appl. Energy*, vol. 314, May 2022, Art. no. 118828.

[31] L. Zhao, J. Li, Q. Li, and F. Li, "A federated learning framework for detecting false data injection attacks in solar farms," *IEEE Trans. Power Electron.*, vol. 37, no. 3, pp. 2496–2501, Mar. 2022.

[32] Y. Li, X. Wei, Y. Li, Z. Dong, and M. Shahidehpour, "Detection of false data injection attacks in smart grid: A secure federated deep learning approach," *IEEE Trans. Smart Grid*, vol. 13, no. 6, pp. 4862–4872, Nov. 2022.

[33] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[34] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1999, pp. 223–238.

[35] R. Deng, G. Xiao, R. Lu, H. Liang, and A. V. Vasilakos, "False data injection on state estimation in power systems—Attacks, impacts, and defense: A survey," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 411–423, Apr. 2017.

[36] X. Yin, Y. Zhu, and J. Hu, "A subgrid-oriented privacy-preserving microservice framework based on deep neural network for false data injection attack detection in smart grids," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 1957–1967, Mar. 2022.

[37] M. De Cock, R. Dowsley, A. C. A. Nascimento, D. Railsback, J. Shen, and A. Todoki, "High performance logistic regression for privacy-preserving genome analysis," *BMC Med. Genomics*, vol. 14, no. 1, pp. 1–18, Dec. 2021.

[38] Y. Lindell, "How to simulate it—A tutorial on the simulation proof technique," in *Tutorials on the Foundations of Cryptography* (Information Security and Cryptography), Y. Lindell, Ed. Cham, Switzerland: Springer, 2017. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-57048-8_6.

[39] S. Meinecke et al., "SimBench—A benchmark dataset of electric power systems to compare innovative solutions based on power flow analysis," *Energies*, vol. 13, no. 12, p. 3290, Jun. 2020.

Hong-Yen Tran is currently pursuing the Ph.D. degree with the School of Engineering and IT, The University of New South Wales Canberra at ADFA, Canberra, Australia. Her research interests are in the field of secure and verifiable computation, applied cryptography in cyber-physical systems, and bio-cryptography.

Jiankun Hu (Senior Member, IEEE) is currently a Professor with the School of Engineering and IT, The University of New South Wales Canberra at ADFA, Canberra, Australia. He is also an invited Expert of Australia Attorney-General's Office, assisting the draft of Australia National Identity Management Policy. He has received nine Australian Research Council (ARC) Grants and has served at the Panel on Mathematics, Information, and Computing Sciences, Australian Research Council ERA—The Excellence in Research for Australia Evaluation Committee in 2012. His research interests are in the field of cyber security covering intrusion detection, sensor key management, and biometrics authentication. He has many publications in top venues, including the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, *Pattern Recognition*, and the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. He is a Senior Area Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY.

Xuefei Yin received the B.S. degree from Liaoning University, Liaoning, China, the M.E. degree from Tianjin University, Tianjin, China, and the Ph.D. degree from The University of New South Wales Canberra at ADFA, Canberra, Australia. He is currently with the School of Information and Communication Technology, Griffith University, Gold Coast, QLD, Australia. He has published articles in top journals, including IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, *ACM Computing Surveys*, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and IEEE INTERNET OF THINGS JOURNAL. His research interests include biometrics, pattern recognition, privacy-preserving, and intrusion detection.

Hemanshu R. Pota received the B.E. degree from the Sardar Vallabhbhai Regional College of Engineering and Technology, Surat, India, in 1979, the M.E. degree from the Indian Institute of Science, Bengaluru, India, in 1981, and the Ph.D. degree from The University of Newcastle, NSW, Australia, in 1985, all in electrical engineering. He is currently an Associate Professor with The University of New South Wales Canberra at ADFA, Canberra, Australia. He has held visiting appointments with Columbia University, New York City, NY, USA; the University of California at Los Angeles, Los Angeles; the University of Delaware; Iowa State University; Kansas State University; Old Dominion University; the University of California at San Diego, San Diego; and the Centre for AI and Robotics, Bengaluru.