applied sciences

MDPI

Article

An Improved Federated Learning-Assisted Data Aggregation Scheme for Smart Grids

Bo Pang [1], Hui-Hui Liang [1], Ling-Hao Zhang [1], Yu-Fei Teng [1], Zheng-Wei Chang [1], Ze-Wei Liu [2,*], Chun-Qiang Hu [2] and Wen-Hao Mou [2]

[1] State Grid Sichuan Electric Power Research Institute, Chengdu 610059, China; pang-bo@outlook.com (B.P.); bance2011@163.com (H.-H.L.); zlh.nju@gmail.com (L.-H.Z.); yfteng@163.com (Y.-F.T.); changzw@126.com (Z.-W.C.)

[2] School of Big Data & Software Engineering, Chongqing University, Chongqing 400030, China; chu@cqu.edu.cn (C.-Q.H.); mwh@cqu.edu.cn(W.-H.M.)

* Correspondence: zwliu@cqu.edu.cn

Abstract: In the context of rapid advancements in artificial intelligence (AI) technology, new technologies, such as federated learning and edge computing, have been widely applied in the power Internet of Things (PIoT). When compared to the traditional centralized training approach, conventional federated learning (FL) significantly enhances privacy protection. Nonetheless, the approach poses privacy concerns, such as inferring other users' training data through the global model or user-transferred parameters. In light of these challenges, this research paper introduces a novel privacy-preserving data aggregation scheme for the smart grid, bolstered by an improved FL technique. The secure multiparty computation (SMC) and differential privacy (DP) are skillfully combined with FL to combat inference attacks during both the learning process and output inference stages, thus furnishing robust privacy assurances. Through this approach, a trusted third party can securely acquire model parameters from power data holders and securely update the global model in an aggregated way. Moreover, the proposed secure aggregation scheme, as demonstrated through security analysis, achieves secure and reliable data aggregation in the electric PIoT environment. Finally, the experimental analysis shows that the proposed scheme effectively performs federated learning tasks, achieving good model accuracy and shorter execution times.

Keywords: federated learning; privacy-preserving; smart grid; SMC; DP; aggregation

check for updates

Citation: Pang, B.; Liang, H.-H.; Zhang, L.-H.; Teng, Y.-F.; Chang, Z.-W.; Liu, Z.-W.; Hu, C.-Q.; Mou, W.-H. An Improved Federated Learning-Assisted Data Aggregation Scheme for Smart Grids. *Appl. Sci.* **2023**, *13*, 9813. https://doi.org/10.3390/app13179813

Academic Editor: Subhas Mukhopadhyay

Received: 30 June 2023
Revised: 4 August 2023
Accepted: 26 August 2023
Published: 30 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Over the past few years, the global economy has undergone a substantial period of rapid development. However, this growth has been accompanied by increasing energy demands, expanding resource supplies, and intensifying environmental pollution [1]. In response to these pressing challenges, the power Internet of Things (PIoT) technology has been steadily addressing existing issues. PIoT capitalizes on its sensitive transmission capability, real-time communication prowess, intelligent control capacity, and robust information security capability. As a result, it seamlessly integrates traditional Internet of Things (IoT) concepts and transforms into a cutting-edge system for efficiently operating and managing power systems [2,3]. This concept fully utilizes the mature advantages of IoT technology in areas such as smart sensors, wide-area communication, and edge computing. Resource coordination is accomplished via intelligent network control and grid distribution.

The application of PIoT technology spans across diverse sectors, encompassing industrial production, smart cities, agricultural production, environmental monitoring, security surveillance, and home automation, achieving significant economic benefits [4]. However, PIoT also faces security issues. Due to its structure and operational characteristics, it demands high-security protection to ensure the confidentiality, integrity, and availability of

data, meeting the requirements of information security. It is more complex than conventional IoT as it involves a greater variety of devices and denser node deployment. Therefore, the overall security risks of PIoT are also higher. Additionally, the health condition of power equipment is a crucial factor in ensuring the secure and reliable operation of the power system [5]. In the power grid, the interconnection between terminal devices and edge nodes allows for the comprehensive and flexible perception of electricity consumption status, enabling intelligent sensing of real-time operational conditions and digitalized maintenance. This facilitates the efficient processing of collected information and prompt intelligent decision-making. For power companies, important tasks encompass monitoring and the intelligent management of power equipment status, and key aspects include the collection, storage, and processing of large amounts of power equipment data [6]. Therefore, applying machine learning technology to the management, analysis, and decision-making of power equipment data can effectively enhance the efficiency of related work in PIoT.

Existing applications of machine learning and PIoT typically involve training large-scale machine learning models by uploading local data from edge nodes to cloud servers. Data management, analysis, and decision-making are conducted through centralized approaches based on cloud storage [7]. In traditional power systems, direct data transmission occurs between edge nodes and cloud servers. However, this method of data transmission is vulnerable to attacks by malicious third parties, resulting in data breaches that compromise the security of power data and lead to significant power incidents [8]. Moreover, future trends will require real-time data privacy and security to reduce the risk of data leaks and enhance the overall reliability of the power grid. Centralized data centers no longer suffice in addressing the confidentiality, authenticity, integrity, and reliability requirements of the information security of power IoT data.

Federated learning, as a machine learning approach, involves multiple data owners and a central server collaborating to train models [9]. It enables the maintenance of multiple distributed training datasets simultaneously. Federated learning offers the opportunity to employ more sophisticated models while simultaneously safeguarding privacy and the security of local data. It facilitates the collaborative training of machine learning models among several data owners without the necessity of transmitting data to a central server. The benefits of federated learning stem from the fact that the raw data remain locally stored and are not exchanged or transmitted, replacing the conventional approach of aggregating all data directly to achieve training goals [10].

1.1. Challenge

Due to the mutual exclusivity among multiple edge nodes in the smart grid of Internet of Things (IoT), data sharing between these edge nodes becomes challenging, affecting the data aggregation capability in the smart grid of IoT. To address this issue, the federated learning (FL) framework enables data sharing among multiple edge nodes, ensuring effective data aggregation in the power sector.

While federated learning is the way to go with training models in privacy-dependant systems, it does pose some problems that need to be addressed. One of the biggest concerns is data privacy. Firstly, attackers may infer original data by analyzing the shared parameters (parameter inversion). Even without direct access to raw data, they could deduce sensitive information by observing updates to the shared model's parameters. Secondly, in some cases, the shared model might contain data labeling or partial data, leading to the potential leakage of sensitive information. Lastly, if the parameter updates lack noise, attackers may infer individual data by monitoring the gradient updates, thus compromising privacy.

Consequently, we propose an enhanced FL system that is specifically designed for data aggregation in the smart grid domain. The secure multi-party computation (SMC) [11] and differential privacy (DP) [12] is integrated with FL to mitigate inference attacks during the learning process and output process, thereby ensuring ample privacy guarantees. Within this system, a trusted third party can securely acquire model parameters from data holders and securely update the global model through aggregation. Throughout this

entire process, the integrity of data from each party is maintained, ensuring that private information cannot be inferred from the global model [13].

1.2. Contributions

This paper makes the following contributions:

(1) As a pioneering work, this paper presents an FL-based framework for the PIoT. Building upon the existing system architecture of the smart grid of IoT and the collaborative cloud–edge-end structure, the framework is enhanced to address the current privacy issues and leverage the characteristics of federated learning.

In this framework, machine learning tasks are executed at edge nodes, and only the model's trained parameters are transmitted to the cloud server. This approach ensures the privacy and security of local data, as no data from any party involved in the learning process are leaked when compared to directly aggregating all power consumption data on the cloud server. The use of the cloud server for aggregating parameter data transmitted from edge nodes involves identity verification during the aggregation process, effectively guarding against forged identities of edge nodes within the framework, thus enhancing overall security.

(2) A secure aggregation algorithm based on federated learning is proposed. This method improves upon traditional federated learning approaches by combining secure multi-party computations and differential privacy techniques. By securely aggregating data, this method prevents attackers from decrypting the encrypted data during the learning process and inferring private information from intermediate results, thus providing sufficient privacy protection for the aggregation of data in the PIoT.

Moreover, this method utilizes aggregation algorithms based on secret sharing and key negotiations to support edge node dropouts, mid-process, while ensuring the security of data transmission. Through security analysis, it has been demonstrated that the proposed secure aggregation algorithm in this paper can achieve secure and reliable data aggregation from edge nodes to cloud servers in the PIoT.

(3) A privacy-preserving aggregation result verification algorithm is proposed. This method utilizes homomorphic hash functions and pseudo-random functions to protect data privacy while supporting edge nodes in verifying the aggregation results on the cloud server, ensuring the data correctness during the federated learning process. The performance analysis of the efficiency and overall overhead of the federated learning process demonstrates that the framework and method proposed in this paper can effectively execute FL tasks without compromising the normal efficiency of data model training.

(4) A performance evaluation concerning the efficiency and overall cost of the FL process demonstrates that the proposed framework and methods can effectively execute federated learning tasks with good model accuracy and a shorter learning time.

The subsequent sections of this paper are structured in the following manner. Section 2 examines previous studies on privacy-preserving strategies in smart grids. Our suggested model is introduced in Section 3. Section 4 presents the preliminaries relevant to our scheme. In Sections 5 and 6, we present the technical intuition and proposed scheme in detail. The security analysis is displayed in Section 7. The experiments and performance analysis are presented in detail in Section 8. Finally, a brief conclusion is drawn in Section 9.

2. Related Works

We will conduct a comparative analysis of the existing work based on the features within our system. In reference [14], Shokri and Shmatikov introduced a technique aimed at training neural networks on horizontally partitioned data by exchanging update parameters. This method ensures the privacy of participants' training data while maintaining the accuracy of the resulting model. Additionally, Phong et al. [15] presented a novel deep learning system that employs additive techniques to safeguard gradients from inquisitive servers, thus preserving user privacy, while assuming no degradation in the accuracy of deep learning with homomorphic encryption. Cao et al. [16] proposed an optimized

federated learning framework that incorporates a balance between local differential privacy, data utility, and resource consumption. They classified users based on varying levels of privacy requirements and provided stronger privacy protection for sensitive users. Moreover, Bonawitz et al. [17] devised a secure aggregation approach for high-dimensional data and proposed a federated learning protocol that supports offline users using secure multi-party computing.

Aono et al. [18] proposed a secure framework using homomorphic encryption to protect training data in logistic regression. Their approach ensures security even when dealing with large datasets. Nikolaenko et al. [19] utilized homomorphic encryption and Yao's scrambled circuit to construct a privacy-preserving system for ridge regression. This system outputs the best-fit curve without exposing the input data to additional information. Truex et al. [20] suggested a method that combines DP and SMC to address inference threats and generate highly accurate models for federated learning techniques. Hu et al. [21] proposed a privacy-preserving approach that ensures that user data satisfy DP and can be effectively learned for distributed user data. Chase [22] combined DP and SMC with machine computing, presenting a feasible protocol for learning neural networks in a collaborative manner while protecting the privacy of each recording.

In contrast to prevailing methods, our system primarily enhances the FL process's security by integrating advanced techniques, such as homomorphic encryption and DP. By doing so, it effectively safeguards against inference threats and ensures the privacy of users' sensitive data, even if they decide to withdraw from the learning process before completion.

3. System Model and Design Goals

We specify the system model and present the design goals in this part.

3.1. System Model

As described in Figure 1, there are primarily four entities: trusted authority (TA), smart meter (SM), edge node (denoted as user u_i), and cloud server (CS).

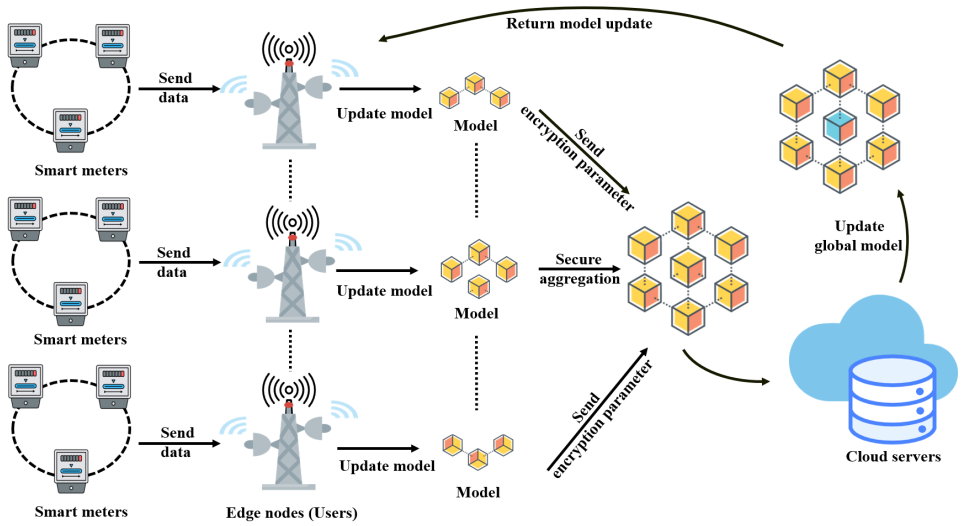


Figure 1. The improved federated learning-assisted data aggregation model.

- Trusted authority: The TA is a trusted entity with key generation and parameter configuration capabilities. It generates and assigns public and private keys for each node, and initializes system parameters for subsequent node data processing.
- Smart meter: It is a terminal device that directly connects to users and collects all electricity-related data online. The primary responsibility of a smart meter is to sense and gather information that reflects the operational status of the power system. The collected data include electricity consumption data, power data, peak and off-peak electricity price data, and communication information, among others. The collection

and analysis of these data can assist grid managers and consumers in gaining a better understanding of electricity usage, optimizing electricity consumption strategies, reducing waste, and enhancing the overall efficiency and reliability of the power system. The smart meter itself has limited storage resources and computing capabilities. Therefore, it needs to transmit data to the corresponding edge nodes for storage and model training. However, the data held by the smart meter are highly sensitive and require encryption using appropriate algorithms to guarantee the security of the transmission procedure.

- Edge node: As the data holder (denoted as user u_i), it possesses certain data storage and computing capabilities and performs calculations on the collected data from smart devices within a specific area. In order to ensure that the device data collected throughout the learning process remain local, the edge node needs to use local data to execute machine learning algorithms for model training. Then, the parameters of the trained model are transmitted to the cloud server to engage in the subsequent phases of the learning process. During the learning process, the edge node is responsible for communicating with other edge nodes and the cloud server, collaboratively executing secure aggregation algorithms, and providing support to protect the parameter data from decryption and inference during transmission.
- Cloud server: It constitutes a pivotal component of the framework, possessing essential storage and computing capabilities. Its primary function revolves around serving as the central data processing center, effectively connecting with all relevant edge nodes situated in the respective areas. It collaborates with the edge nodes to execute secure aggregation algorithms, decrypts the encrypted data, eliminates added noise, and calculates new parameters for training the global model. Additionally, it is responsible for storing and utilizing the new global model. Moreover, the cloud server must verify the identity of every edge node it interacts with.

3.2. Design Goals

- Privacy: This involves protecting the privacy and security of data transmitted from smart devices to edge nodes, ensuring that the data remain local throughout the entire learning process. Only encrypted model parameters are directly transmitted between the u_i and CS. The data are protected by secure aggregation algorithms during transmission, making them resistant to third-party attacks, including interception or the inference of results. Each edge node is only aware of its own transmitted data, and during interactions with other nodes and the cloud server, only obtains fragmentary values processed by secret sharing algorithms. The cloud server serves the purpose of aggregating data and training the global model, but it cannot obtain the original parameters and training data of the nodes through the acquired parameters, and it cannot infer the data of other nodes through collusion with any single node.
- Identity authentication: Ensures that attackers cannot launch attacks on the system by forging identities. During the learning process, if attackers successfully launch attacks by using forged identities or data, it may result in incorrect aggregation results or even the leakage of original data. The utilization of zero-knowledge proof in this framework enables the cloud server to authenticate the identities of communication nodes. This feature allows for the detection of attackers who attempt to forge identities and it safeguards the data from tampering, thereby enhancing the security of the learning process.
- Fault tolerance: Ensures that the failure of any edge node during the FL does not affect the correctness of the aggregation results.
- Verifiability: Once the aggregation results are calculated, it is necessary for the CS to transmit the results back to the edge nodes that participated. Throughout this process, the edge nodes have the ability to verify the accuracy of the returned results and reject any results that cannot be verified.

4. Preliminaries

We shall initially present the fundamental principles of FL, followed by an overview of the cryptographic source languages and modules employed in this framework.

4.1. Federated Learning

In federated learning, multiple participants can collaboratively train a model without the need to centralize their original datasets. Instead, each participant retains their own data and conducts model training locally. By employing techniques such as encryption and secure computation, participants can aggregate the model's update information to achieve global performance improvement while preserving the privacy of individual data [23]. Federated learning enables the sharing of knowledge among multiple organizations or individuals, promoting the development and application of machine learning under the premise of sensitive data protection.

There are two primary aspects of privacy in the context of learning: the privacy during the learning process [24], and the privacy of the learning outcome [25]. Privacy in the learning process refers to the process in which participants and the server transmit data between each other, and one of the participants infers the private data of the other participants from the data transmitted by the server. The privacy of the learning outcome involves the disclosure of intermediate data transmitted by participants and the server or the disclosure of the model (M) that the server ultimately obtains.

Privacy in the learning process: In an FL environment, participant P_i ought to transmit the parameters of the local learning model x_i to the server, and if an attacker obtains x_i, it is possible to infer P_i's private data D_i. Therefore, the risk of inferring x_i must be considered in a privacy-protected federated learning system.

To address this potential risk, secure multi-party computation (SMC) is a widely adopted approach. In SMC, the security model inherently involves multiple participants and provides robust security proof within a well-defined simulation framework. This framework ensures "absolute zero knowledge", meaning that each participant remains completely unaware of any information other than their own inputs and outputs. Zero-knowledge is highly desirable due to its enhanced data privacy and security features. However, achieving such properties often requires the implementation of more complex computational protocols, which might not always be practical. Under certain circumstances, accepting partial knowledge disclosure might be considered acceptable, as long as adequate security assurances are provided. While SMC effectively mitigates risks during the learning process, it is important to recognize that federated learning systems relying solely on SMC may still be susceptible to learning outcome inferences. Consequently, privacy-protecting FL systems must also account for potential inferences related to the learning outcomes. This entails incorporating additional measures or protocols to prevent unauthorized access or inference of sensitive information regarding the learning process results.

Privacy of learning outcomes: Learning outcomes encompass the intermediate outcomes attained throughout the federated learning process as well as the ultimate learning model. Several studies have demonstrated that an attacker can deduce information about the training data [25]. Consequently, in a federated learning environment, the inference made regarding the learning outcome must adhere to the principle that the owner $\mathcal{U}_i$ of the data D_i is the only party capable of inferring any outcome related to D_i.

The differential privacy (DP) framework is commonly employed to address privacy concerns pertaining to output results [26]. By introducing noise to the data through differential privacy, applying techniques like k-anonymity or diversification, or by utilizing inductive methods to obfuscate sensitive attributes, the data become indistinguishable, safeguarding user privacy. Nonetheless, these approaches often necessitate the transfer of data to external entities, resulting in a trade-off between accuracy and privacy.

4.2. Differential Privacy

Differential privacy (DP) was introduced by Cynthia Dwork at Microsoft Research Labs and was initially designed for tabular data. It provides a statistical concept of privacy, allowing for highly accurate privacy-preserving statistical analysis of sensitive datasets [27]. The level of DP is measured using privacy loss parameters (ϵ, δ), where smaller values of (ϵ, δ) correspond to higher privacy levels. Strictly speaking, for all $S \subseteq Range(A)$ and neighboring datasets D and D', if the randomization algorithm A satisfies the following equation, it is considered (ϵ, δ)-differentially private [28]:

$$P(A(D) \in S) \leq e^{\epsilon} P(A(D') \in S) + \delta \tag{1}$$

In order to fulfill the requirements of DP, it is necessary to introduce noise into the output of the algorithm. Gaussian mechanisms are adopted in our scheme [29].

4.3. Secret Sharing

Secret sharing is a sophisticated technique used to protect sensitive information by breaking it into multiple fragments and distributing those fragments to various entities, thereby concealing the original secret. The process of secret sharing involves allocating portions of the secret value to different individuals, and the shared value is obtained by summing up the contributions from each participant. Consequently, each party only gains access to a limited portion of the shared value, ensuring that no single entity possesses the complete secret. Depending on the specific scenario, it may be necessary to retrieve all or a predefined number of shared values to reconstruct the original secret. Shamir's secret sharing is an exemplary method that relies on the polynomial equation theory to achieve a high level of information security. Additionally, it effectively utilizes matrix operations, leading to enhanced computational efficiency and accelerated processing [30]. This approach is widely recognized for its robustness and practicality in safeguarding confidential data.

The secure aggregation algorithm proposed in this paper uses secret sharing based on Shamir's secret sharing, which allows users to decompose their desired secret s into n fragments and distribute them to n other individuals for safekeeping. In this way, when the number of collected secret fragments is less than a threshold t, no one can obtain information about the secret s. However, when t individuals present their own secret fragments, the secret s can be reconstructed. The secret sharing comprises two processes: the sharing process, executed by the algorithm $\textbf{SS.share}(s, t, \mathcal{U}) \rightarrow (u, s_u)_{u \in U}$, produces a collection of secret fragments s_u for each user u involved in the sharing. These fragments are uniquely associated with each user. Similarly, the recombination process, performed by the algorithm $\textbf{SS.recon}((u, s_u)_{u \in V}, t) \rightarrow s$, combines the fragments $(u, su)_{u \in V}$ and generates the reconstructed secret s.

4.4. Key Agreement

The Diffie–Hellman key agreement [31] is adopted in our scheme, which comprises three algorithms: **KA.param** is responsible for initialization, **KA.gen** is responsible for generating key pairs, and **KA.agree** is used for key negotiation. The aforementioned agreement keys are utilized for the exchange of information among users throughout the FL process.

4.5. Threshold Homomorphic Encryption

Threshold homomorphic encryption is a cryptographic scheme that allows for performing computations on encrypted data without decrypting it. It incorporates the concept of threshold cryptography, where multiple participants collectively hold shares of a secret key [32]. The encryption scheme ensures that a certain threshold number of participants must collaborate to perform operations on the encrypted data. In threshold homomorphic encryption, encrypted values can be combined through specific homomorphic operations,

such as addition or multiplication, while ensuring the secrecy of the original data remains intact. The encrypted result of the computation can be decrypted by the participants only when the threshold number of participants collaborate to collectively decrypt it. This cryptographic scheme provides a secure and distributed way to perform computations on sensitive data while preserving privacy. It allows for collaborative computation without revealing individual input and ensures that no single participant has complete access to the decrypted data [33].

4.6. Homomorphic Hash Functions

The message (M) undergoes encryption using a homomorphic hash function that is resistant to collisions [34,35].

$$HF(M_i) = (A_i, B_i) = (g^{HF_{\delta,\rho}(M_i)}, h^{HF_{\delta,\rho}(M_i)}) \tag{2}$$

In this case, both δ and ρ are randomly chosen secret keys from $\mathbb{Z}_q$. The homomorphic hash function possesses the following properties:

1. $HF(M_1 + M_2) \leftarrow (g^{HF_{\delta,\rho}(M_1)+HF_{\delta,\rho}(M_2)}, h^{HF_{\delta,\rho}(M_1)+HF_{\delta,\rho}(M_2)})$.
2. $HF(\alpha M_1) \leftarrow (g^{\alpha HF_{\delta,\rho}(M_1)}, h^{\alpha HF_{\delta,\rho}(M_1)})$.

Additional properties of this homomorphic hash function are documented in [34,35].

4.7. Pseudorandom Functions

We utilize the pseudorandom functions [36] in this paper. The pseudorandom function PF_K uses the secret key K to generate some arguments for the subsequent verification section:

$$PF_K(I_1, I_2) = (E, F) = (g^{\gamma I_1 \gamma I_2 + \nu I_1 \nu I_2}, h^{\gamma I_1 \gamma I_2 + \nu I_1 \nu I_2}) \tag{3}$$

In our scheme, the combination of a pseudo-random function and a homomorphic hash function is used to authenticate the aggregated results provided by the system [37].

5. Technical Intuition

In this model, each user $u \in \mathcal{U}$ possesses a private vector x_u of size m, x_u includes the parameters of the model obtained by the user after local training using private data. We further assume that $\sum_{u\in\mathcal{U}} x_u$ and its elements are for R in $\mathbb{Z}_R$. The server (S) plays the role of an aggregator, which aggregates the data transmitted by each user (u) and trains the data to obtain a new model M. The objective of this model is to ensure the privacy of the user's data during the FL process. The server only obtains the sum of the parameters provided by the users to the client $\sum_{u\in\mathcal{U}} x_u$, while each user has the ability to verify the final model returned by the CS.

In order to provide privacy to the model parameters x_u of the users, we add two random vectors, $s_{u,v}$ and b_u, to each user's x_u to compute x_u in perfect secrecy. We assume that the list of users is ordered and that there is a random vector $s_{u,v}$ between each pair of users (u, v). Before a user sends x_u to the server, the random vector $s_{u,v}$ between u and every other user v is added to x_u, namely:

$$y_u = x_u + \sum_{v\in U, u<v} s_{u,v} - \sum_{v\in U, u>v} s_{v,u} \tag{4}$$

and, finally, the server calculates:

$$X = \sum_{u\in U} y_u = \sum_{u\in\mathcal{U}} (x_u + \sum_{v\in\mathcal{U}, u<v} s_{u,v} - \sum_{v\in\mathcal{U}, u>v} s_{v,u}) = \sum_{u\in\mathcal{U}} x_u \tag{5}$$

To ensure that a user cannot quit midway before committing y_u to the server, leading to the server's inability to recover from X, we implemented a Pseudorandom Generator (**PRG**) technique. This method involves generating random vectors using **PRG** [36,37], and then sending the seeds used for generating these random vectors to all other users

through threshold secret sharing of Diffie–Hellman secret shares. By doing so, even if a user v decides to quit midway, the server can still reconstruct a random vector associated with v using the shares provided by other users. And to prevent the server from obtaining the private data x_u of user u by asking users for their shares of $s_{u,v}$, we add another random vector $\mathbf{PRG}(b_u)$ to x_u, where b_u is seeded to a random vector that is unique to each user. Also, transferring these seeds for **PRG** between users saves more communication overhead than transferring the entire random vector. Then, before a user sends x_u to the server, the user needs to calculate the following:

$$y_u = x_u + p_u + \sum_{v \in \mathcal{U}_2} p_{u,v} \tag{6}$$

and the server needs to calculate

$$X = \sum_{u \in \mathcal{U}'} x_u = \sum_{u \in U'} y_u - \sum_{u \in U'} p_u + \sum_{u \in U', v \setminus U'} p_{v,u} \tag{7}$$

In the above formula, $\mathcal{U}'$ means all surviving users, and $p_{u,v} = \Delta_{u,v} \cdot \mathbf{PRG}(s_{u,v})$, where $\Delta_{u,v} = 1$ when $u > v$, $\Delta_{u,v} = -1$ when $u < v$, $p_u = \mathbf{PRG}(b_u)$.

Prior to the secure aggregation round, the server needs to make a specific decision for each user u. The server can request either $s_{u,v}$ or b_u shares, which are associated with u from each remaining user other than u. It is important to note that an honest user will never disclose both shares of the same user. Once the server has collected at least t shares of $s_{u,v}$ from the departing user and t shares of b_u from all remaining users, it can subtract the remaining masks to obtain the precise aggregation result.

6. Proposed Scheme

We will provide an overview of the implementation of our FL system, focusing on addressing four key privacy concerns in the FL environment. Firstly, we aim to safeguard the privacy of local data throughout the FL process. Secondly, we are committed to preventing any potential disclosure of intermediate results or the final model. Furthermore, we guarantee the integrity of results even if a user exits prematurely. Lastly, we incorporate mechanisms that enable users to authenticate the results provided by the *CS*.

The initial phase of the scenario is shown in Figure 2. Figure 3 shows the interaction diagram between the edge node and the cloud server after the initialization of the scheme. The system is structured around six interaction rounds, designed to accomplish the aforementioned objectives. Initially, the system initiates and generates all the necessary keys for both users and *CS*. Subsequently, each user P_i encrypts their gradient x_u and transmits it to the *CS*. Upon receiving an adequate number of messages from all active users, *CS* aggregates the gradients from each user and returns the computed results, along with a corresponding Proof, to each user. Finally, each user verifies the Proof to determine whether to accept or reject the computed result. The process then loops back to Round 0 to commence a new iteration. Figure 4 shows the specific process of interaction between edge nodes and cloud servers.

6.1. Share Decryption and Share Combining

In order to safeguard the confidentiality of the local data throughout the learning process, we employed a threshold homomorphic encryption technique to encrypt the model parameters of each user u. Additionally, to mitigate the risk of information leakage pertaining to intermediate learning results and the final model, Gaussian noise is incorporated into the encryption process of the model parameters y_u. Furthermore, to establish the authenticity of user identities, a Fiat–Shamir zero-knowledge proof [38] is constructed, enabling the server to authenticate the user's identity.

Initialization phase

- **Preparation Stage**
  - Information provided to all users: secret sharing threshold t, public parameter pp generated by **KA.param**(k), parameters m, number of users n and R of input space $\mathbb{Z}_{n2}^{*}$, secret sharing space F. There is a secure channel between each user and the server, and they share a public value a_u.
- **Round 0: Advertise Keys**

  User u :
  - Key pairs (c_u^{PK}, c_u^{SK}), (s_u^{PK}, s_u^{SK}), (h_u^{PK}, h_u^{SK}) are generated by **KA.gen**(pp), which are used to encrypt and decrypt transmission content, secret sharing, and limited homomorphic encryption.
  - Send $(c_u^{PK}||s_u^{PK}||h_u^{PK})$ to the server.

  Server :
  - Collect the information of at least t users and mark this group of users as $\mathcal{U}_1$.
  - Broadcast $\{(v, c_v^{PK}, s_v^{PK}, h_v^{PK})\}_{v \in \mathcal{U}_1}$ to all users in $\mathcal{U}_1$.

Figure 2. Initialization process.

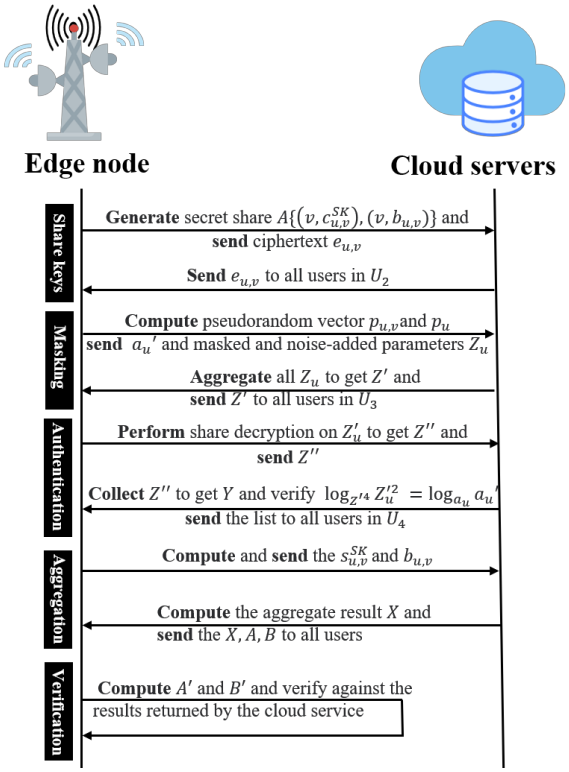


Figure 3. Interaction between the edge node and the cloud server.

During Round 2, each user utilizes their individual public key h_u^{PK} to encrypt the previously noise-added data y_u, resulting in

$$Z_u = \mathbf{Enc}_{pk}(y_u + noise(\epsilon, t'))$$

where ϵ represents the privacy guarantee, and t' refers to the number of non-colluding parties.

Federated learning-assisted aggregation and verification-specific process

- **Round 1: Share Keys**
  User u :
  - Receive the Broadcast $\{(v, c_v^{PK}, s_v^{PK}, h_v^{PK})\}_{v\in\mathcal{U}_1}$ from the server.
  - Take a random number b_u in $\mathbb{F}$ as the seed of **PRG**.
  - Generate a shares $\{(v, s_{u,v}^{SK})\}_{v\in\mathcal{U}_1} \leftarrow$ **SS.share**$(s_u^{SK}, t, \mathcal{U}_1)$ on s_u^{SK} with a threshold of t in $\mathcal{U}_1$.
  - Generate a shares $\{(v, b_{u,v})\}_{v\in\mathcal{U}_1} \leftarrow$ **SS.share**$(b_u, t, \mathcal{U}_1)$ on s_u^{SK} with a threshold of t in $\mathcal{U}_1$.
  - For each of the remaining users $v \in \mathcal{U}_1\backslash\{u\}$, calculate $e_{u,v} \leftarrow$ **AE.enc**$($**KA.agree**$(c_u^{SK}, c_v^{PK}), u||v||s_{u,v}^{SK}||b_{u,v})$. **KA.agree**$(c_u^{SK}, c_v^{PK})$ is the agreement key between u and v, $(u||v||s_{u,v}^{SK}||b_{u,v})$ is the content to be encrypted.
  - Send all calculated ciphertext $e_{u,v}$ to the server.

  Server :
  - Collect the ciphertext of at least t users and mark this group of users as $\mathcal{U}_2$.
  - Send $\{e_{u,v}\}_{v\in\mathcal{U}_2}$ to all users in $\mathcal{U}_2$.
- **Round 2: Masking**
  User u :
  - Receive ciphertext $\{e_{u,v}\}_{v\in\mathcal{U}_2}$ from the server.
  - For each of the remaining users $v \in \mathcal{U}_2\backslash u$,calculate the agreement key $s_{u,v} \leftarrow$ **KA.agree**(s_u^{SK}, s_v^{PK}) and the pseudorandom $p_{u,v} = \Delta_{u,v} \cdot$ **PRG**$(s_{u,v})$, where $\Delta_{u,v} = 1$ when $u > v$, and $\Delta_{u,v} = -1$ when $u < v$.
  - Calculate the private mask $p_u =$ **PRG**(b_u),$y_u = x_u + p_u + \sum_{v\in\mathcal{U}_2} p_{u,v}$ *mod* R and $Z_u =$ **Enc**$_{pk}(y_u + noise(\epsilon, t'))$.
  - Calculate $a'_u = a^{\Delta f(u)}$ *mod* n^2,$f(x) = \sum_u^{k-1} x^u$ *mod* nm, $HF(x_u) = (A_u, B_u) = (g^{HF_{\delta,\rho}(x_u)}, h^{HF_{\delta,\rho}(x_u)})$,$\delta$, $PF_{K_1}(n) = (\gamma_n, \nu_n)$, $PF_{K_2}(\tau) = (\gamma, \nu)$, $PF_K(n, \tau) = (E_n, F_n) = (g^{\gamma_n\gamma}, h^{\nu_n\nu})$, where $\tau = sum$, $L_n = (E_n \cdot A_n^{-1})^{1/d}$, $Q_n = (F_n \cdot B_n^{-1})^{1/d}$, and ρ are random keys in $\mathbb{Z}_q$, and g and h are generators of G_1 and $G2$ in $\mathbb{Z}_q$.
  - Send the masked and noise-added parameters Z_u , a'_u, A_n, B_n, L_n, Q_n, $\Omega_n = 1$ to the server.

  Server :
  - Collect messages of at least t users and mark this group of users as $\mathcal{U}_3$, then send user list $\mathcal{U}_3$ to users in $\mathcal{U}_3$.
  - Aggregate all Z_u to obtain $Z' =$ **Enc**$_{pk}(y_1 + y_2 + \cdots + y_n) \leftarrow Z_1 \circ Z_2 \circ \cdots \circ Z_n$.
  - Send Z' to all users in $\mathcal{U}_3$.
- **Round 3: User Authentication**
  User u :
  - Receive Z' , then compute $Z'_u = Z'^{2\Delta f(u)}$ and perform share decryption $Z'' = Dec_{sk}(Z')$.
  - Send Z'', Z'_u to the server.

  Server :
  - Collect Z'', Z'_u of at least t users.
  - Proof that $\log_{Z'^4}(Z'^2_u) = \log_{a_u}(a'_u)$.
  - Obtain share combining $Y = y_1 + y_2 + \cdots + y_n$, then mark this group of users as $\mathcal{U}_4$.
  - Send the list to all users in $\mathcal{U}_4$.
- **Round 4: Secure aggregation**
  User u :
  - Receive the list $\{v\}_{v\in\mathcal{U}_4}$ and decrypt the ciphertext $v' \parallel u' \parallel s_{v,u}^{SK} \parallel b_{v,u} \leftarrow AE.dec($**KA.agree**$(c_u^{SK}, c_v^{PK}), e_{v,u})$ for other users $v \in \mathcal{U}_2\backslash\{u\}$.
  - Send $s_{v,u}^{SK}(v \in \mathcal{U}_2\backslash\mathcal{U}_4)$ and $b_{v,u}(v \in \mathcal{U}_4)$ to the server.

  Server :
  - Collect at least t replies, and mark this group of users as $\mathcal{U}_5$.
  - Reconstruct $S_u^{SK} \leftarrow$ **SS.recon**$(\{s_{u,v}^{SK}\}_{v\in\mathcal{U}_4}, t)$ for each user $u \in \mathcal{U}_2\backslash\mathcal{U}_4$, and use it to recompute $P_{v,u}$ for all $v \in \mathcal{U}_4$ using the **PRG**.
  - Reconstruct $b_u \leftarrow$ **SS.recon**$(\{b_{u,v}\}_{v\in\mathcal{U}_5}, t)$
  - Aggregate and output $X = \sum_{u\in\mathcal{U}_4} x_u = \sum_{u\in\mathcal{U}_4} y_u - \sum_{u\in\mathcal{U}_4} p_u + \sum_{u\in\mathcal{U}_4, v\in U2\backslash\mathcal{U}_4} p_{v,u}$, compute $A = \prod_{n=1}^{n=|\mathcal{U}_4|} A_n$, $B = \prod_{n=1}^{n=|\mathcal{U}_4|} B_n$, $L = \prod_{n=1}^{n=|\mathcal{U}_4|} L_n$, $Q = \prod_{n=1}^{n=|\mathcal{U}_4|} Q_n$, $\Omega = \prod_{n=1}^{n=|\mathcal{U}_4|} \Omega_n$.
- **Round 5: Correctness verification**
  User u :
  - Receive X and (A, B) from the server, then compute $(A', B') = (g^{HF_{\delta,\rho}(X)}, h^{HF_{\delta,\rho}(X)})$, $\varphi = \sum_{n\in U_4}(\gamma_n\gamma + \nu_n\nu)$ and $\phi = e(g, h)^{\varphi}$.
  - Verify whether $(A, B) = (A', B')$, $e(A, h) = e(g, B)$, $e(L, h) = e(g, Q)$, $\phi = e(A, h) \cdot e(L, h)$. If the validation of each equation is successful, then the aggregation returned by the server is correct, i.e., $X = x_1 + x_2 + \cdots + x_n$.

Figure 4. Federated learning-assisted aggregation and verification-specific process.

Based on the Paillier cryptosystem, each user is required to compute the following expression:

$$Z_u = g^{(y_u + noise(\epsilon, t')} r^n$$

where $n = p \cdot q$, and $r \in \mathbb{Z}^*_{n^2}$ denotes a randomly chosen number.

In Round 3, $\bar{t} = n - t' + 1$ users are selected by the server to perform decryption sharing on Z', which is the service aggregate.

Each user initially computes

$$a'_u = a^{\Delta f(u)} \ mod \ n^2 \ , \qquad f(x) = \sum_{u}^{k-1} x^u$$

and transmits it to the server. Upon receiving these values, the server computes the aggregated encrypted value Z' as $Z' = \mathbf{Enc}pk(y1 + y_2 + \cdots + y_n) \leftarrow Z_1 \circ Z_2 \circ \cdots \circ Z_n$. In other words, Z' is equal to $g^Y r^n$, where $Y = \sum_{u \in \mathcal{U}_3} y_u$.

Subsequently, each user computes $Z'_u = Z'^{2\Delta f(u)}$ based on the received Z' and forwards it to the server. Upon receiving these data, the server verifies the correctness as follows:

$$\begin{aligned} \log_{Z'^4}(Z'^2_u) &= \log_{Z'^4}(Z'^{4\Delta f(u)}) \\ &= \log_{a_u}(a^{\Delta f(u)}) \\ &= \log_{a_u}(a'_u) \end{aligned}$$

If the server possesses a sufficient number of shares and valid **proof**, it can merge them to obtain the final result by selecting a subset S of $\bar{t}$ shares, combining them as follows:

$$M = \prod_{u \in S} Z'^{2\lambda^S_{0,u}}_u \ mod \ n^2$$

where $\lambda^S_{0,u} = \Delta \prod_{u' \in S ß} \frac{-u}{u-u'} \in \mathbb{Z}$.

The value of M can be expressed as $M = Z'^{4\Delta^2} \mod n^2$. It should be noted that $4\Delta^2 = 0 \mod \lambda$. Therefore, the server can deduce that $M = (1+n)^{4\Delta^2 Y} \mod n^2$. Considering that $(1+n)^x \equiv 1 + nx \mod n^2$, we have $1 + 4n\Delta^2 Y$. The server can calculate $Y = \frac{M-1}{4n\Delta^2}$.

6.2. Verification

In Round 5, each user receives (A, B) and verifies its correctness through proofing:

$$\begin{aligned} (A, B) &= (\prod_{n=1}^{n=\mathcal{U}} A_n, \prod_{n=1}^{n=\mathcal{U}} B_n) \\ &= (g^{\sum_{n=\mathcal{U}} HF_{\delta,\rho}(x_n)}, h^{\sum_{n=\mathcal{U}} HF_{\delta,\rho}(x_n)}) \\ &= (g^{HF_{\delta,\rho}(\sum_{n=\mathcal{U}} x_n)}, h^{HF_{\delta,\rho}(\sum_{n=\mathcal{U}} x_n)}) \\ &= (A', B') \\ e(A, h) &= e(g^{HF_{\delta,\rho}(\sigma)}, h) = e(g, h^{HF_{\delta,\rho}(\sigma)}) \\ &= e(g, B) \\ e(L, h) &= e(g^{\sum_{n \in \mathcal{U}_4}(\gamma_n \gamma + \nu_n \nu) - HF_{\delta,\rho}(x_n)}, h)^{1/d} \\ &= e(g, h^{\sum_{n \in \mathcal{U}_4}(\gamma_n \gamma + \nu_n \nu) - HF_{\delta,\rho}(x_n)})^{1/d} \\ &= e(g, Q) \\ & e(A, h) \cdot e(L, h)^d \\ &= e(g^{HF_{\delta,\rho}(\sigma)}, h) \cdot e(g^{\sum_{n \in \mathcal{U}_4}(\gamma_n \gamma + \nu_n \nu) - HF_{\delta,\rho}(x_n)}, h) \\ &= e(g, h)^{\sum_{n \in \mathcal{U}_4}(\gamma_n \gamma + \nu_n \nu)} \\ &= \Phi \end{aligned}$$

6.3. Reducing Noise with SMC

The SMC framework is harnessed to efficiently address the noise introduced by differential privacy (DP). Let σ_s denote the noise parameters in the federated learning (FL) algorithm, while S_s represents the sensitivity of the allocated budget. Within this framework, the noise is effectively reduced by a factor of $(t-1)$, leading to enhanced data privacy and accuracy.

Since $t-1 < n$, the noise present in the aggregated value satisfies the requirements of DP strictly. Additionally, as the SMC framework involves a maximum of $\bar{t}$ colluders, it is not possible to decrypt the private data of other users. To summarize, the SMC framework effectively reduces the noise introduced by DP, thereby ensuring the security of the FL system.

7. Security Analysis

7.1. Correctness Verification

In Round 5, each remaining user receives the final values of σ, A, B, L, Q sent by the server. The user verifies the correctness of $\phi = e(A,h)\dot{e}(L,h)^d$ by applying the l-BDHI assumption [35]. If ϕ is verified correctly, the user can deduce A and L and further verify that $e(A,h) = e(g,B)$, $e(L,h) = e(g,Q)$ based on the DDH assumption [39]. By performing these verifications, the user can confirm that the server has computed the correct values of B and Q. This implies that the result X returned by the server is accurate.

The proof process can be performed easily using l-BDHI [35] and DDH [39]. The specific steps of the proof process are not included here.

7.2. Threshold Homomorphic Encryption

During Round 2 and Round 3, each user transmits a'_u, Z'_u to the server, and the server calculates Z'. Subsequently, the server conducts the zero-knowledge proof to authenticate that $\log_{Z'^4}(Z'^2_u) = \log_{a_u}(a'_u)$. As every user shares the exclusive public value a_u with the server in the setup phase, this assures the server that the user's identity has not been counterfeited by an adversary.

The non-interactive zero-knowledge proof described above employs the Fiat–Shamir heuristic, which can be readily derived from the relevant publication [38].

7.3. Honest But Curious Security

In this section, the technical principles used in this system for security purposes will be described. Before the user transmits the parameters, we use **PRG** to add pairs of random vectors to the data, which hide the specific information entered by the user, so that the final result obtained by the server is always uniformly random to the result obtained without adding the random vectors.

Within our system, the transfer of user data occurs in an environment where all participants are honest but curious. During this process, the users collaboratively engage in computational logic while preserving the utmost confidentiality of their private data. Each participant gains access solely to the computation results, completely oblivious to the data of other participants or the intermediate outcomes of the computation process. This ensures a high level of privacy and security for all involved parties, including both the participants and the server. The system effectively shields the sensitive data from being disclosed, fostering trust and confidentiality among the users, while allowing them to collectively obtain the required results without any compromise on their privacy.

We assume that the server S interacts with a group of n users, denoted as $\mathcal{U}$, and the threshold is set to t. Users have the freedom to exit at any point during their interaction with the server. Let $\mathcal{U}_i$ represent the set of users who have sent data to the server in Round $i-1$ or, in other words, the users who have not exited by Round $i-1$. Therefore, we have $\mathcal{U} \supseteq \mathcal{U}_1 \supseteq \mathcal{U}_2 \supseteq \mathcal{U}_3 \supseteq \mathcal{U}_4 \supseteq \mathcal{U}_5$. $\mathcal{U}_2 \backslash \mathcal{U}_3$ refers to the users who sent data to the server in Round 1 but did not send data within the time limit of Round 2. For any arbitrary subset C, $\mathrm{REALC}^{\mathcal{U},t,k}(x_U, \mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3, \mathcal{U}_4, \mathcal{U}_5)$ is a random variable representing a collective

perspective of the parties in C during the execution of the system. In the following, two theorems applied in this system will be given. In a user honest-but-curious environment, the joint view with fewer than t users can only infer its own data. In the server's honest-but-curious environment, the server and the joint view with fewer than t users can only infer the sum of its own data and the other users' data.

Theorem 1. *chochasasassasass*(*Against Multiple Users*) *For all* k, t, $\mathcal{U}$, xu, $\mathcal{U}_1$, $\mathcal{U}_2$, $\mathcal{U}_3$, $\mathcal{U}_4$, $\mathcal{U}_5$, C *with* $t <= |\mathcal{U}|$, *and* $C \supseteq \mathcal{U}, \mathcal{U} \supseteq \mathcal{U}_1 \supseteq \mathcal{U}_2 \supseteq \mathcal{U}_3 \supseteq \mathcal{U}_4 \supseteq \mathcal{U}_5$, *there is a PPT simulator* SIM. *The output of* SIM *is identical to* $\text{REAL}_C^{U,t,k}$:

$$\text{REAL}_C^{\mathcal{U},t,k}(x_{\mathcal{U}}, \mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3, \mathcal{U}_4, \mathcal{U}_5) \equiv \text{SIM}_C^{\mathcal{U},t,k}(x_C, \mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3, \mathcal{U}_4, \mathcal{U}_5)$$

Proof of Theorem 1. The collective perspective of users in C relies only on their own inputs, so it is possible to run their inputs through the simulator and obtain a simulated perspective of users in C that is consistent with REAL. The server's response in Round 2 solely consists of a list of user identities without disclosing specific values of yu. As a result, the simulator can execute the inputs of users in C and utilize hypothetical values to represent the inputs of all honest participants not in C. This ensures that the collective perspective of users in C in the simulation remains consistent with REAL. □

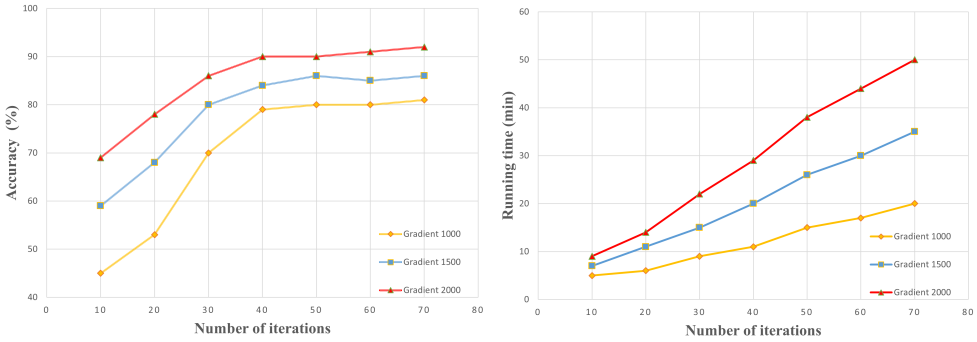
By combining the insights from the previous hybrid and the arguments presented above, we are able to define a PPT simulator, denoted as SIM. The purpose of this simulator is to mimic the behavior of the system and produce outputs that are computationally indistinguishable from the outputs of the real system, denoted as REAL.

8. Evaluation

We conducted an assessment of the honest-but-curious version of our system and provided a comprehensive overview of its performance in various aspects. Our scheme offers user validation of aggregated results, ensuring the privacy of users' sensitive data throughout and after the execution process. Additionally, users have the flexibility to exit the system while ensuring the integrity and accuracy of the final outcome at any stage of the system's operation. Through our evaluation, we aimed to demonstrate the effectiveness and reliability of our system, highlighting its capabilities in maintaining data security, supporting user validation, and accommodating user exits while preserving result correctness.

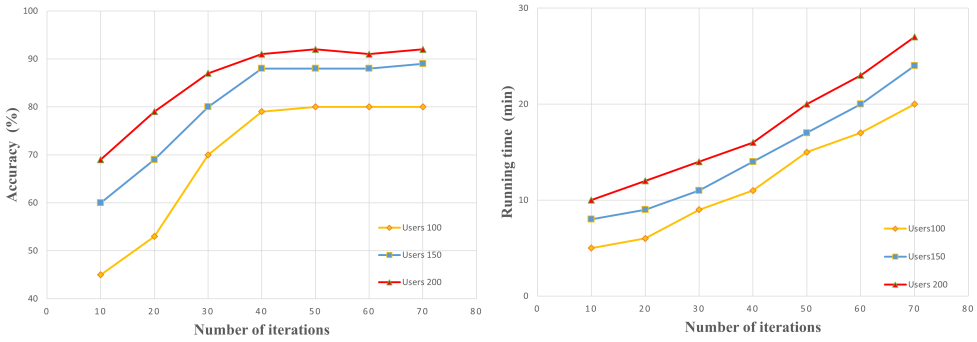
Considering that the accuracy of the model in federated learning (FL) depends on two crucial factors—the number of participating users during training and the magnitude of the local gradient held by each user—we conducted our experiments with careful attention to detail. We meticulously recorded the user count and gradient size in each experiment, aiming to thoroughly examine the correlation between these variables, the model's accuracy, and the system's overhead.

Figure 5 presents the classification accuracy and execution time in relation to various gradients in our experimental trials. The system's computation overhead and accuracy exhibit variations contingent upon the number of users and the magnitude of the gradient. As illustrated in Figure 5a, an augmentation in the number of local gradients possessed by users corresponds to an elevated accuracy in the model's outcome. However, it also results in greater computation overhead per round. Once the number of gradients surpasses a certain threshold, the model's accuracy tends to stabilize. Likewise, Figure 6 demonstrates that solely considering changes in the number of users, an elevation in user count enhances the computation overhead while improving the model's accuracy. An increase in both the number of users and gradients unavoidably amplifies the computation overhead. Consequently, it is advisable to select an appropriate number of users and gradients, taking into account the computational overhead, rather than simply striving for more.



(**a**) Classification accuracy (**b**) Running overhead

Figure 5. No dropout, $|\mathcal{U}| = 100$.



(**a**) Classification accuracy (**b**) Running overhead

Figure 6. No dropout, $|\mathcal{G}| = 1000$.

Figure 7 depicts the experimental evaluation of the accuracy of results obtained from the user verification server. During the verification stage, each user is tasked with validating the aggregated results received from the server and making a decision to either accept or reject them before updating their local model. Our analysis involves a diverse set of users, and we compare their verified server aggregation results with the ground truth. As illustrated in the figure, the validated data precisely align with the actual data, underscoring the confidence in the accuracy of the aggregated results generated by the server after undergoing the rigorous validation process. This validates the effectiveness of our approach, as the verification stage ensures the reliability of the server's aggregated outcomes and confirms that they remain consistent with the ground truth.

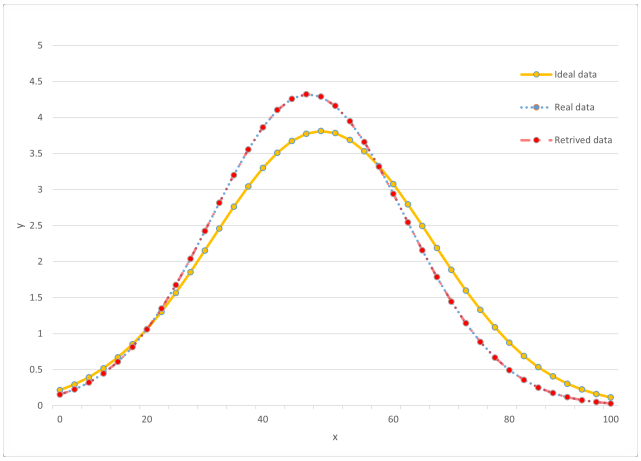
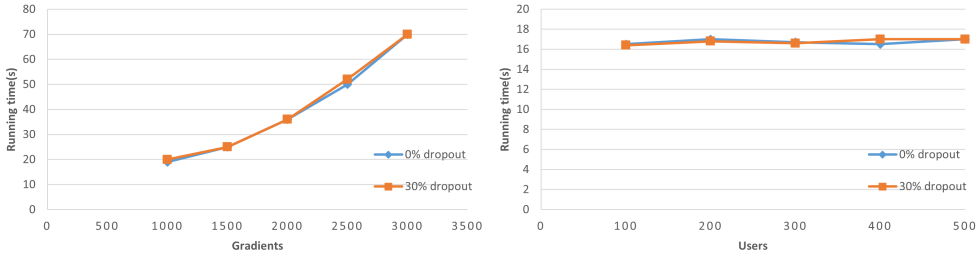


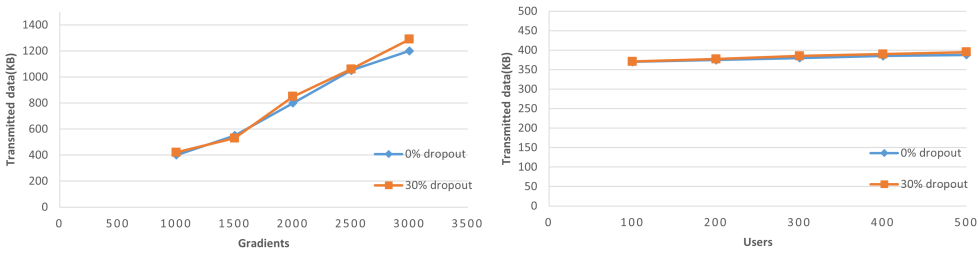
Figure 7. Verification accuracy.

Figures 8 and 9 provide insight into the computation and communication overhead incurred by users during the execution of the system, measured in terms of runtime and data volume. As depicted in Figure 8, it is apparent that the computation overhead escalates as the number of local gradients possessed by users increases. Nevertheless, if only the number of users increases without any change in the local gradients, the computation overhead remains constant. These overheads primarily relate to the number of user gradients, meaning that an increase in the number of user gradients leads to an overall rise in the total overhead. Figure 9 demonstrates that the communication overhead of users follows a similar pattern as the computation overhead. Regardless of any variations in the number of users, the transmitted data volume demonstrates a linear increase as the number of user gradients rises. Additionally, Table 1 reveals that the communication overhead of users within the system primarily stems from the masking and verification stages, where Num refers to the number of clients, DP means dropout, SK stands for share keys, MI denotes masking, UA marks user authentication, SA represents secure aggregation, and CR denotes correctness verification.



(a) $|\mathcal{U}| = 100$ **(b)** $|\mathcal{G}| = 1000$

Figure 8. Total running overhead for each user.



(a) $|\mathcal{U}| = 100$ **(b)** $|\mathcal{G}| = 1000$

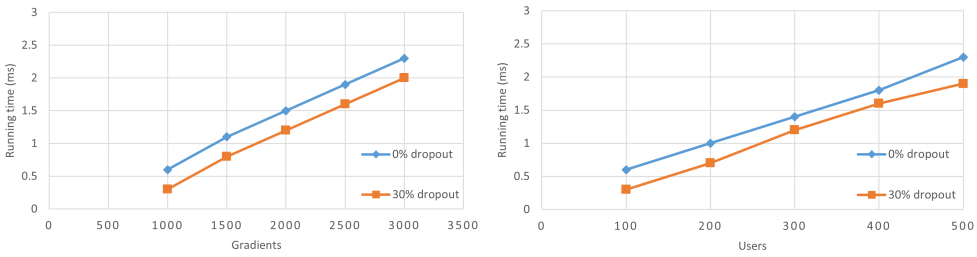
Figure 9. Total transmitted data for each user.

Table 1. Communication overhead in each stage (ms).

	Client	**Server**	**Server**	**Client**	**Server**	**Server**
Num	100	100	100	300	300	300
DP	0%	0%	30%	0%	0%	30%
AK	100	1	1	100	5	5
SK	1502	203	198	13,638	347	325
MI	4250	560	524	7425	821	789
UA	980	248	244	1452	274	252
SA	532	1432	9648	841	2680	23,167
CR	6712	0	0	8521	0	0
Total	14,076	2444	10615	21,977	4127	24,538

Figure 10 presents a visualization of the computation overhead experienced by the server throughout the system's execution. Contrary to the variation pattern observed in the computation overhead of users, the server exhibits a linear correlation with both the number of local gradients held by users and the number of users. As the number

of gradients or users increases, the server's computation overhead rises correspondingly. Furthermore, Table 1 reveals that the server's computation overhead is greatly influenced by the number of users who withdraw from the system. Overall, the primary sources of overhead for the CS during system execution are the aggregation of gradients transmitted by users and the removal of masks.



(a) $|\mathcal{U}| = 100$ **(b)** $|\mathcal{G}| = 1000$

Figure 10. Total running overhead of the server in the verification process.

The algorithm proposed in this paper and its comparison with existing federated learning algorithms in terms of functionality are shown in Table 2. The comparison indicators include transmission process protection (TPP), intermediate results protection (IRP), support for exit (SE), identity recognition (IR) , and results verification (RV). In the table, the presence of "Y" indicates the availability of the feature in the corresponding scheme, while "N" denotes its absence. All algorithms in this table achieve transmission process protection in federated learning. Bonawitz et al.'s algorithm, for the first time in the federated learning framework, introduces a privacy-preserving approach by adding random vectors to transmitted data to protect data privacy and using digital signature technology for identity identification. Truex et al.'s algorithm uses homomorphic encryption to protect the aggregation intermediate results in the federated learning environment and achieves identity verification through zero-knowledge proof algorithms. Xu et al.'s algorithm provides authentication for the correctness of aggregation results while protecting transmitted data and implementing identity identification using hash functions. Cao et al.'s algorithm combines federated learning with differential privacy, making it difficult for attackers to infer the original data, thus protecting the intermediate results of federated learning. This table shows that our proposed algorithm successfully achieves the aforementioned functionalities, affirming the efficacy of our approach.

Table 2. Comparison of federated learning schemes.

Scheme	TPP	IRP	SE	IR	RV
Cao's scheme [16]	Y	Y	N	N	N
Truex's scheme [20]	Y	Y	N	Y	N
Bonawitz's scheme [40]	Y	N	Y	Y	N
Xu's scheme [41]	Y	N	N	Y	N
Oue's scheme	Y	Y	Y	Y	Y

Figure 11 presents a comparison between the proposed approach and the federated learning algorithm in terms of time consumption. In the comparative experiment, the federated learning environment was uniformly configured with the possibility of the mid-way node dropout. In our proposed approach, additional phases were introduced compared to the algorithms presented by Truex and Xu et al. These extra phases include support for mid-way node dropout, validation result correctness, and protection of intermediate results. Notably, the validation phase requires higher time consumption. Consequently, from the graph, it can be observed that our proposed approach, which includes the validation phase, takes longer execution time compared to Truex et al.'s approach, which lacks

a correctness validation phase, regardless of the number of nodes. Regarding Xu et al.'s approach, which also incorporates a validation phase, our proposed approach exhibits similar time consumption when the number of nodes is relatively small. However, as the number of nodes increases, our proposed approach maintains a shorter execution time compared to Xu et al.'s approach.

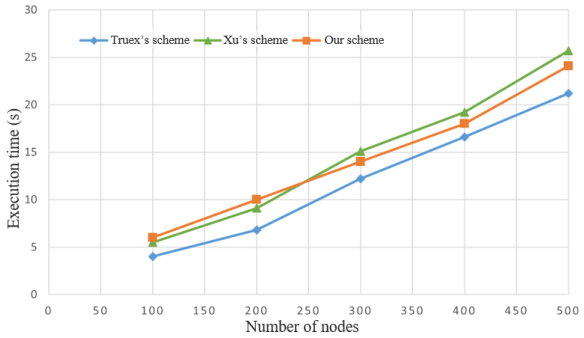


Figure 11. Total running time comparison.

Figure 12 is configured with the possibility of mid-way node dropout, and the number of nodes is fixed at 200. From the graph, it is evident that the federated learning algorithm executed in our proposed approach consistently maintains higher model accuracy across various iterations compared to the other two schemes.

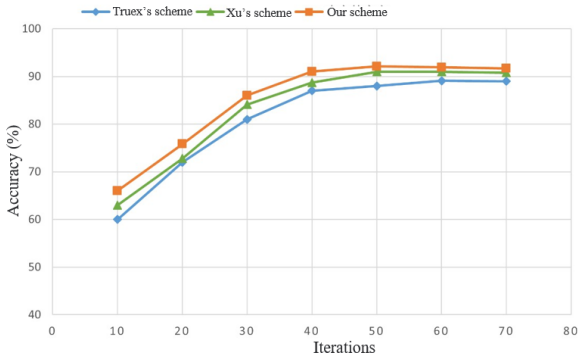


Figure 12. Accuracy comparison.

Through the above experimental analysis, it can be seen that the proposed method can provide more accurate data aggregation and, thus, obtain better overall model performance.

9. Conclusions

This paper introduces a federated learning system designed to strengthen the privacy of learning processes and outcomes in smart grids through the implementation of SMC and DP techniques. Moreover, our system offers support for edge nodes by incorporating secret sharing and key protocols for early exit. Additionally, user verification of server aggregation results is facilitated through the use of homomorphic hash functions and pseudo-random functions. The conducted security analysis showcases the robustness of the scheme in providing comprehensive support for edge nodes. Finally, the performance evaluation conducted with real data demonstrates the exceptional performance of our system throughout the FL process.

Author Contributions: Conceptualization, B.P. and Z.-W.L.; methodology, C.-Q.H.; software, H.-H.L.; validation, L.-H.Z. and Y.-F.T.; formal analysis, Z.-W.C.; investigation, Z.-W.L. and W.-H.M.; resources, C.-Q.H.; data curation, H.-H.L.; writing—original draft preparation, Z.-W.L.; writing—review and editing, B.P.; visualization, H.-H.L.; supervision, L.-H.Z.; project administration, Y.-F.T.; funding acquisition, Y.-F.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the State Grid Sichuan Electric Power Company's science and technology project (SGSCDK00LYJS2200130).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, X.; Sun, M.; Ma, Y.; Zhang, L.; Zhang, Y.; Yang, R.; Liu, Q. Using Sensor Network for Tracing and Locating Air Pollution Sources. *IEEE Sens. J.* **2021**, *21*, 12162–12170. [CrossRef]
2. Miyanabe, K.; Gama Rodrigues, T.; Lee, Y.; Nishiyama, H.; Kato, N. An Internet of Things Traffic-Based Power Saving Scheme in Cloud-Radio Access Network. *IEEE Internet Things J.* **2019**, *6*, 3087–3096. [CrossRef]
3. Li, Y.; Cheng, X.; Cao, Y.; Wang, D.; Yang, L. Smart Choice for the Smart Grid: Narrowband Internet of Things (NB-IoT). *IEEE Internet Things J.* **2018**, *5*, 1505–1515. [CrossRef]
4. Liu, Y.; Chi, C.; Zhang, Y.; Tang, T. Identification and Resolution for Industrial Internet: Architecture and Key Technology. *IEEE Internet Things J.* **2022**, *9*, 16780–16794. [CrossRef]
5. Wenjing, W.; Jianing, X.; Xin, S.; Zhihui, L.; Yingqi, Y.; Xinhe, W. The research on correlation of electrical power system stability in high proportion wind power area. In Proceedings of the 2021 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, 29–31 July 2021; pp. 563–567. [CrossRef]
6. Shi, Y.; Liu, Z.; Hu, C.; Cai, B.; Xie, B. An Efficient and Secure Power Data Trading Scheme Based on Blockchain. In Proceedings of the Wireless Algorithms, Systems, and Applications—16th International Conference, WASA 2021, Nanjing, China, 25–27 June 2021; Proceedings, Part I; Liu, Z., Wu, F., Das, S.K., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2021; Volume 12937, pp. 147–158. ._12. [CrossRef]
7. Bae, M.; Kim, K.; Kim, H. Preserving privacy and efficiency in data communication and aggregation for AMI network. *J. Netw. Comput. Appl.* **2016**, *59*, 333–344. .: 10.1016/j.jnca.2015.07.005. [CrossRef]
8. Yao, S.; Yang, X.; Song, Z.; Yang, X.; Duan, D.; Yang, H. Maze Routing: An Information Privacy-aware Secure Routing in Internet of Things for Smart Grid. In Proceedings of the 2022 7th International Conference on Communication, Image and Signal Processing (CCISP), Chengdu, China, 18–20 November 2022; pp. 461–465. [CrossRef]
9. Mills, J.; Hu, J.; Min, G. Multi-Task Federated Learning for Personalised Deep Neural Networks in Edge Computing. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 630–641. [CrossRef]
10. Mou, W.; Fu, C.; Lei, Y.; Hu, C. A Verifiable Federated Learning Scheme Based on Secure Multi-party Computation. In Proceedings of the Wireless Algorithms, Systems, and Applications—16th International Conference, WASA 2021, Nanjing, China, 25–27 June 2021; Proceedings, Part II; Liu, Z., Wu, F., Das, S.K., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2021; Volume 12938, pp. 198–209. ._16. [CrossRef]
11. Dong, C.; Weng, J.; Liu, J.N.; Yang, A.; Liu, Z.; Yang, Y.; Ma, J. Maliciously Secure and Efficient Large-Scale Genome-Wide Association Study With Multi-Party Computation. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 1243–1257. [CrossRef]
12. Liu, Z.; Hu, C.; Xia, H.; Xiang, T.; Wang, B.; Chen, J. SPDTS: A Differential Privacy-Based Blockchain Scheme for Secure Power Data Trading. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 5196–5207. [CrossRef]
13. Wang, B.; Hu, C.; Liu, Z. A Secure Aggregation Scheme for Model Update in Federated Learning. In Proceedings of the Wireless Algorithms, Systems, and Applications—17th International Conference, WASA 2022, Dalian, China, 24–26 November 2022; Proceedings, Part I; Wang, L., Segal, M., Chen, J., Qiu, T., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2022; Volume 13471, pp. 500–512. ._41. [CrossRef]
14. Shokri, R.; Shmatikov, V. Privacy-Preserving Deep Learning. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15), New York, NY, USA, 12–16 October 2015; pp. 1310–1321. [CrossRef]
15. Phong, L.T.; Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1333–1345. [CrossRef]
16. Cao, H.; Liu, S.; Zhao, R.; Xiong, X. IFed: A novel federated learning framework for local differential privacy in Power Internet of Things. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 1–18. [CrossRef]
17. 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 (CCS '17), New York, NY, USA, 30 October–3 November 2017; pp. 1175–1191. [CrossRef]
18. Aono, Y.; Hayashi, T.; Trieu Phong, L.; Wang, L. Scalable and secure logistic regression via homomorphic encryption. In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, 9–11 March 2016; pp. 142–144.
19. Nikolaenko, V.; Weinsberg, U.; Ioannidis, S.; Joye, M.; Boneh, D.; Taft, N. Privacy-preserving ridge regression on hundreds of millions of records. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 19–22 May 2013; pp. 334–348.

20. Truex, S.; Baracaldo, N.; Anwar, A.; Steinke, T.; Ludwig, H.; Zhang, R.; Zhou, Y. A hybrid approach to privacy-preserving federated learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, London, UK, 15 November 2019; pp. 1–11.
21. Hu, R.; Guo, Y.; Li, H.; Pei, Q.; Gong, Y. Personalized Federated Learning with Differential Privacy. *IEEE Internet Things J.* **2020**, *7*, 9530–9539. [CrossRef]
22. Chase, M.; Gilad-Bachrach, R.; Laine, K.; Lauter, K.E.; Rindal, P. Private Collaborative Neural Network Learning. *IACR Cryptol. EPrint Arch.* **2017**, *2017*, 762.
23. Agrawal, R.; Srikant, R. Privacy-preserving data mining. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 16–18 May 2000; pp. 439–450.
24. Nasr, M.; Shokri, R.; Houmansadr, A. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv* **2018**, arXiv:1812.00910.
25. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership inference attacks against machine learning models. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–24 May 2017; pp. 3–18.
26. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Hofburg Palace, Austria, 25–27 October 2016; pp. 308–318.
27. Dwork, C. *International Conference on Theory and Applications of Models of Computation*; Inproceedings; Springer: Berlin/Heidelberg, Germany, 2008.
28. Dwork, C. Differential privacy: A survey of results. In Proceedings of the International Conference on Theory and Applications of Models of Computation, Xi'an, China, 25–29 April 2008; pp. 1–19.
29. Dwork, C.; Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* **2014**, *9*, 211–407. [CrossRef]
30. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
31. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]
32. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; pp. 223–238.
33. Damgård, I.; Jurik, M. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In Proceedings of the International Workshop on Public Key Cryptography, Cheju Island, Republic of Korea, 13–15 February 2001; pp. 119–136.
34. Yun, A.; Cheon, J.H.; Kim, Y. On homomorphic signatures for network coding. *IEEE Trans. Comput.* **2010**, *59*, 1295–1296. [CrossRef]
35. Fiore, D.; Gennaro, R.; Pastro, V. Efficiently verifiable computation on encrypted data. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 844–855.
36. Blum, M.; Micali, S. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM J. Comput.* **1984**, *13*, 850–864. [CrossRef]
37. Yao, A.C. Theory and application of trapdoor functions. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982), Chicago, IL, USA, 3–5 November 1982; pp. 80–91.
38. Shoup, V. Practical threshold signatures. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Bruges, Belgium, 14–18 May 2000; pp. 207–220.
39. Boneh, D.; Franklin, M. Identity-based encryption from the Weil pairing. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; pp. 213–229.
40. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Federated Learning on User-Held Data. *arXiv* **2016**, arXiv:1611.04482.
41. Xu, G.; Li, H.; Liu, S.; Yang, K.; Lin, X. VerifyNet: Secure and Verifiable Federated Learning. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 911–926. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.