Privacy-Enhanced Federated Learning with Weighted Aggregation

Jiale Guo[1](✉), Ziyao Liu[1](✉), Kwok-Yan Lam[1], Jun Zhao[1], and Yiqiang Chen[2]

[1] Nanyang Technological University, Singapore, Singapore
{jiale001,ziyao002}@e.ntu.edu.sg, {kwokyan.lam,junzhao}@ntu.edu.sg
[2] University of Chinese Academy of Sciences and Peng Cheng Laboratory, Beijing, China
yqchen@ict.ac.cn

Abstract. The pervasive adoption of machine learning (ML) techniques by social network operators has led to a growing concern in the personal data privacy of their customers. ML inevitably accesses and processes users' personal data, which could potentially breach the relevant privacy protection regulations if not performed carefully. In this backdrop, Federated Learning (FL) is an emerging area that allows ML on distributed data without the data leaving their stored location. Typically, FL starts with an initial global model, with each datastore uses its local data to compute the gradient based on the global model, and uploads their gradients (instead of the data) to an aggregation server, at which the global model is updated and then distributed to the local datastores iteratively. However, depending on the nature of the services operated by social networks, data captured at different locations may carry different significance to the business operation, hence a weighted aggregation will be highly desirable for enhancing the quality of the FL model. Furthermore, to prevent the data leakage from aggregated gradients, cryptographic mechanisms are needed to allow secure aggregation of FL. As such, this paper proposes a privacy-enhanced FL scheme, based on cryptographic mechanisms that allow both the data significance evaluation and weighted aggregation of local models in a privacy-preserving manner. Experimental results show that our scheme is practical and secure.

Keywords: Federated learning · Secure aggregation · Data significance evaluation · Homomorphic encryption · Zero-knowledge proof

1 Introduction

Machine learning (ML) has been widely adopted by social network services in order to enhance user experiences and improve revenue opportunities by providing personalized recommendations to the social network users. On the other hand, ML inevitably accesses and processes users' personal data, which could

J. Guo and Z. Liu—Both authors contributed equally to this research.

potentially breach the relevant privacy protection regulations if not performed carefully. The situation is exacerbated by the cloud-based implementation of social network when user data are captured and stored in distributed locations, hence aggregation of the user data for ML could be a serious breach of privacy regulations. Such a scenario where multiple data stores (or custodians) jointly solve a machine learning problem while complying with privacy regulations has attracted tremendous attention from academia and industry. Privacy-preserving techniques such as differential privacy (DP), fully homomorphic encryption (FHE), and secure multi-party computation (MPC) are widely believed to be promising approaches to achieve this goal. However, it is well known that DP requires a tradeoff between data usability and privacy [1], while MPC and HE offer cryptographic privacy with high communication or computation overheads. In this backdrop, Federated Learning (FL) is an emerging area that allows ML on distributed data without the data leaving their stored location [2]. Typically, FL starts with an initial global model, with each data store (in the respective data center) uses its local data to compute the gradient based on the global model, and uploads their gradients (instead of the data) to an aggregation server, at which the global model is updated and then distributed to the local datastores iteratively.

However, depending on the nature of services operated by the social network, data captured at different locations may exhibit disparity. Hence a weighted aggregation scheme will be highly desirable for enhancing the quality of the FL-learned model. In this case, the central server is required to evaluate the significance across all local datasets in order to compute the weightings, and then aggregate all the users' locally trained models according to their weights. To calculate the weight, data significance evaluation (DSE) on data size and data quality are widely adopted. For example, in FedAvg [3], the locally trained models from users are weighted by the percentage of their data size in the total training data. Based on FedAvg, the authors in [4] further evaluate users' label quality by calculating the mutual cross-entropy between data and models of both the central server and users. Although approaches in previous works resulted in improved accuracy of the global FL model, they did not address the privacy guarantee for users' data.

Furthermore, from the angle of privacy protection, social network operators need to address the information leakage from the gradients from which one can derive the users' local training data [5]. Secure[1] aggregation schemes [6,7] aim to deal with this issue. However, during the FL process, the distributed data stores and the central server may still receive fraudulent messages due to insider frauds. For example, a dishonest participant may send fraudulent messages during DSE so as to obtain a manipulated weight. In other cases, a FL participant may upload a fraudulent model to affect the distribution of the global FL model in the current FL round in order to manipulate its weight in the next FL round [8]. Note that such an issue regarding fraudulent messages can be generalized to any FL system.

[1] We use the terms secure and privacy-preserving interchangeably.

In order to address the aforementioned issues, we propose a privacy-enhanced FL scheme with weighted aggregation. To summarize, our contributions are:

– We propose a general privacy-enhanced FL scheme with secure weighted aggregation, which can deal with both the data significance difference, data privacy, and dishonest participants (who send fraudulent messages to manipulate the computed weights) in FL systems.
– We give a detailed example application of our proposed scheme with performance evaluation.
– Compared to existing FL schemes, experimental results show the practicality of our proposed scheme that achieves privacy-enhanced FL with weighted aggregation while providing an additional security guarantee against fraudulent messages with an affordable 1.2 times of run time overheads and 1.3 times communication costs.

Related Works. There have been several schemes proposed to perform privacy-preserving and dropout-resilient aggregation. The main idea of these schemes is to integrate FL with privacy-preserving techniques such as DP, HE, and MPC. For example, the authors in [9] propose a DP-based FL scheme that a level of noise is added to each user's locally trained model before aggregation, which involves the trade-off between model performance and privacy. For HE-based aggregation schemes, each user's locally trained model is encrypted before uploading. Then the central server aggregates the users' model in ciphertext using HE operations and publishes the result for decryption. To deal with dropped participants, the private key is distributed among all participants [10]. Such HE-based schemes incur infeasible overhead for a large-scale FL system as the underlying threshold crypto-system involve expensive protocols. A more efficient and dropout-robust aggregation scheme is based on MPC, as proposed in [6]. In this scheme, each user's locally trained model is masked that the seeds for generating masks are secretly shared among all users using a threshold secret sharing scheme to handle the dropout users. The improved version [7] upon [6] replaces the complete communication graph in [6] with a k-regular graph to further reduce the communication overheads. Another variant TurboAgg [11] divides users into multiple groups and follows a multi-group circular structure for communication. NIKE [12] adopts a non-interactive key exchange protocol relying on non-colluding servers. Nevertheless, all of the schemes mentioned above focus on achieving FL from a privacy-preserving perspective, issues still exist that both the central server and users may send fraudulent messages to each other.

Organisation of the Paper. The rest of the paper is organized as follows. In Sect. 2, we introduce the preliminaries and supporting protocols. In Sect. 3, we describe the rationales for our proposed scheme together with a high-level overview, and the threat model. Then we describe a detailed example of the application of our proposed scheme, and discuss its privacy and security in Sect. 4. Experimental results are presented in Sect. 5 followed by the conclusions in Sect. 6.

2 Preliminaries and Supporting Protocols

This section briefly describes the preliminaries of data significance evaluation, secure aggregation, and related cryptographic protocols used in this work. The participants in our system are divided into two classes: a server S and K users[2] $\mathcal{U} = \{u_1, u_2, \ldots, u_K\}$ that each user $u_i \in \mathcal{U}$ holds a local dataset $\mathcal{D}_i = \{(x_i^j, y_i^j) | j = 1, \ldots, n_i\}$ with size n_i, where x_i^j and y_i^j are the j-th sample and corresponding label in $\mathcal{D}_i$ respectively.

2.1 Data Significance Evaluation

Data significance evaluation scheme is essential for weighted aggregation of users' models in FL. In this work, we adapt the label quality evaluation scheme proposed in [4] to illustrate how to construct an interactive evaluation scheme in a privacy-preserving manner. Note that this scheme can be generalized to any data significance evaluation scheme for federated learning.

By maintaining a small set of benchmark dataset $\mathcal{D}_s$, the central server S is allowed to quantify the credibility C_i of each local dataset $\mathcal{D}_i$ by computing the mutual cross-entropy E_i. In specific, E_i evaluates both the performance of the global model $\mathcal{M}$ on the local dataset $\mathcal{D}_i$, i.e., LL_i, and the performance of the local model of the user $\mathcal{M}_i$ on the benchmark dataset $\mathcal{D}_s$, i.e., LS_i, which is given by:

$$\begin{aligned} E_i &= LS_i + LL_i \\ LS_i &= -\sum_{(x_s, y_s) \in \mathcal{D}_s} y_s \log P\left(y \mid x_s; \mathcal{M}_i\right) \\ LL_i &= -\sum_{(x_u, y_u) \in \mathcal{D}_i} y_u \log P\left(y \mid x_u; \mathcal{M}\right) \end{aligned}$$

Then the weight of user i's model can be defined as: $w_i = \frac{n_i C_i}{\sum_{j=1}^{K} n_j C_i}$ where $C_i = 1 - \frac{e^{\alpha E_i}}{\sum_{j=1}^{K} e^{\alpha E_j}}$. Here α is a hyper-parameter for normalization. These weights can be used in subsequent weighted aggregation to obtain the new global model as $\sum_{i=1}^{K} w_i \mathcal{M}_i$.

2.2 Secure Aggregation

Assume there is a set of client $\mathcal{U}$, and let each client $u \in \mathcal{U}$ holds a vector $\boldsymbol{x}_u$, Secure aggregation scheme [6,7] enables a server S to calculate a sum $z = \sum \boldsymbol{x}_u$ while preserving the privacy of $\boldsymbol{x}_u$. In secure aggregation scheme, a pairwise additive mask is added to $\boldsymbol{x}_u$ (assume a total order on clients) and the client u uploads $\boldsymbol{y}_u$ to the central server instead of $\boldsymbol{x}_u$, i.e. $\boldsymbol{y}_u = \boldsymbol{x}_u + \sum_{u<v} \mathrm{PRG}(\boldsymbol{s}_{u,v}) - \sum_{u>v} \mathrm{PRG}(\boldsymbol{s}_{v,u})$. Here pseudorandom generator (PRG) is able to generate a sequence of random numbers using the seed $\boldsymbol{s}_{u,v}$ which is agreed between client

[2] We use the terms user and client interchangeably.

u and client v. It is straightforward to observe that when aggregating all the $\boldsymbol{x}_u$, the masks will be canceled such that

$$z = \sum_{u \in \mathcal{U}} \boldsymbol{y}_u = \sum_{u \in \mathcal{U}} \left(\boldsymbol{x}_u + \sum_{u<v} \mathrm{PRG}(\boldsymbol{s}_{u,v}) - \sum_{u>v} \mathrm{PRG}(\boldsymbol{s}_{v,u}) \right) = \sum_{u \in \mathcal{U}} \boldsymbol{x}_u$$

Furthermore, the seeds are shared among the clients using standard Shamir secret sharing (n, t) scheme [13] to handle dropout clients. Specifically, we omit some details and summarize the illustrative secure aggregation protocol $z \leftarrow \pi_{SecAgg}(\mathcal{U}, \{\mathcal{M}_i\}, t)$ as follows.

Mask Generation $(\{R_i\}_{u_i \in \mathcal{U}_1}, \mathcal{U}_1) \leftarrow \pi_{MG}(t, \mathcal{U})$. Each user $u_i \in \mathcal{U}$ generates a random matrix R to mask the local model $\mathcal{M}$. The set of alive users after mask generation is denoted as $\mathcal{U}_1$.

Masked Model Aggregation $(y, \mathcal{U}_2) \leftarrow \pi_{MMA}(\{\mathcal{M}\}_{u \in \mathcal{U}_1}, \{R_i\}_{u_i \in \mathcal{U}_1}, \mathcal{U}_1, t)$. Each user $u_i \in \mathcal{U}_1$ computes its masked model $y_u = \mathcal{M} + R$ and uploads it to the server. Then the server collects all masked models from the set of alive users, denoted as $\mathcal{U}_2 \subseteq \mathcal{U}_1$, and computes the sum $y = \sum_{u \in \mathcal{U}_2} y_u$.

Model Aggregation Recovery $z \leftarrow \pi_{MAR}(y, \mathcal{U}_1, \mathcal{U}_2, t)$. The users $u_i \in \mathcal{U}_2$ send information according to $\mathcal{U}_2 \setminus \mathcal{U}_1$ to the server to recover the masks of dropout users $u \in \mathcal{U}_2 \setminus \mathcal{U}_1$. Then the server can compute the sum of models of the alive users $z = \sum_{u_i \in \mathcal{U}_2} \mathcal{M}$.

2.3 Cryptographic Tools

We now introduce the cryptographic tools used in our proposed scheme.

Homomorphic Encryption. A homomorphic crypto-system is a form of encryption scheme that allows computation to be performed on encrypted data directly. It can be denoted as a tuple of algorithms, i.e. $\mathsf{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, where KeyGen is a key generation algorithm, Enc and Dec are used for encryption and decryption, respectively.

In this paper, we adopt Paillier [14] crypto-system which consistsof following algorithms:

- $\mathsf{HE.KeyGen}(p, q)$: For large primes p and q that $gcd(pq, (p-1)(q-1)) = 1$, compute $n = p \cdot q$ and $\lambda = lcm(p-1, q-1)$. Then, select a random integer $g \in \mathbb{Z}^*_{n^2}$ that ensure $gcd(n, L(g^\lambda \bmod n^2)) = 1$, where function L is defined as $L(x) = \frac{x-1}{n}$ and $\lambda = \varphi(n)$. Finally, output a key pair (sk, pk) where the public key is $pk = (n, g)$ and the private key is $sk = (p, q)$.
- $\mathsf{HE.Enc}(pk, m, r)$: For message $m \in \mathbb{Z}_N$, taking the public key pk and a random number $r \in \mathbb{Z}^*_N$, output a ciphertext $c = \mathsf{Enc}(m) = g^m r^n \pmod{n^2}$.
- $\mathsf{HE.Dec}(sk, c)$: For a ciphertext $c < n^2$, taking private key sk, output the plaintext message $m = \mathsf{Dec}(c) = \frac{L(c^\lambda (\bmod n^2))}{L(g^\lambda (\bmod n^2))}$.

Paillier supports two types of operations over ciphertext: (i) homomorphic addition between two ciphertext such that $\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) (\bmod\ n^2) = \mathsf{Enc}(m_1 + m_2 \pmod{n^2})$, and (ii) homomorphic multiplication between a plaintext and a ciphertext such that $(\mathsf{Enc}(m_1))^{m_2} \pmod{n^2} = \mathsf{Enc}(m_1 m_2 \pmod{n})$. For simplicity, we sometimes abuse the notation Enc and Dec instead of $\mathsf{HE.Enc}(pk, m,\ r)$ and $\mathsf{HE.Dec}(sk, c)$ when the context is clear. Note that we denote $\boxplus$ with the homomorphic addition between two ciphertexts, and $\boxtimes$ with the homomorphic multiplication between a ciphertext and a plaintext. In addition, Paillier crypto-system provides semantic security against chosen-plaintext attacks (IND-CPA) such that an adversary will be unable to distinguish pairs of ciphertexts based on the message they encrypt [14].

Zero Knowledge Proof of Plaintext Knowledge. A zero-knowledge proof of plaintext knowledge (ZKPoPK) algorithm enables a prover to prove the knowledge of a plaintext m of some ciphertext $C = Enc(m)$ to a verifier in a given public encryption scheme, without revealing anything about message m. Pailler provides the algorithm $\mathsf{PZKPoPKoZ}$ as given in Algorithm 1 for ZKPoPK of Zero. As such, the prover with a key pair (pk, sk) can prove the knowledge of zero of a ciphertext $u = \mathsf{HE.Enc}(pk, 0, v) \in \mathbb{Z}_{n^2}$ to the honest verifier. The correctness and security proof of Algorithm 1 can be found in [15].

Algorithm 1. Paillier-based ZKPoPK of Zero $b \leftarrow \mathsf{PZKPoPKoZ}(n, u, pk, sk, v)$

Public input: n, u, pk.
Private input of the prover: sk, $v \in \mathbb{Z}_n^*$, such that $u = \mathsf{HE.Enc}(pk, 0, v)$.
Output: β

```
1: The prover randomly chooses $r \in \mathbb{Z}_n^*$ and sends $a = \mathsf{HE.Enc}(pk, 0, r)$ to the verifier.
2: The verifier chooses a random $e$ and sends it to the prover.
3: The prover sends $z = rv^e \bmod n$ to the verifier.
4: The verifier checks that $u, a, z$ are prime to $n$ and that $\mathsf{HE.Enc}(pk, 0, z) = au^e \bmod n^2$. If and only if these requirements satisfy, output $\beta = 1$; otherwise, output $\beta = 0$.
```

3 Proposed Scheme

To evaluate the quality of each user's locally trained model hence the data significance, the standard method is to have the central server keeping a benchmark dataset and having the access to users' models and datasets, which enables computing the weights using some evaluation algorithms [7]. However, a privacy-preserving FL requires that both the user's data and locally trained models cannot be learned by any other participant. As such, the intuitive method is to have the server evaluating the weight using HE based on the encrypted dataset and model of the user. After that, the server and users can jointly aggregate the weighted model using secure aggregation protocol.

In this case, the FL scheme inevitably involves the interaction between the user and the server where participants may send fraudulent messages to manipulate the computed weight. Recall that the user can (i) publish the fraudulent weight (as we use HE for privately computing the weight, thus the decryption is on the user-side, we will explain this in detail later), and (ii) upload the fraudulent locally trained model. Therefore, a verification protocol is needed to guarantee the correctness of both computed weights and users' locally trained models. Our security definition follows the Universal Composability (UC) framework [16]. Active malicious participants can send fraudulent messages. Besides, we assume authenticated channels and signature schemes to ensure the confidentiality and integrity of messages sent by participants.

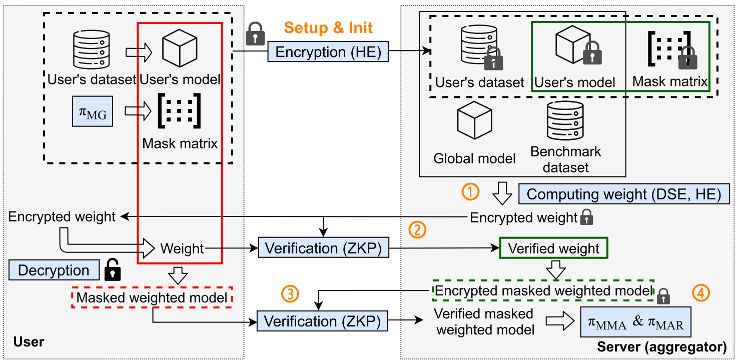


Fig. 1. Privacy-enhanced federated learning scheme with weighted aggregation.

High-Level Overview. As shown in Fig. 1, before proceeding to computing the weight, each user u_i uploads its encrypted dataset Enc($\mathcal{D}_i$), encrypted locally trained model Enc($\mathcal{M}_i$), and encrypted mask matrix Enc(R_i) to the server S. Note that the encrypted datasets are uploaded only once at the very beginning of FL. After that, the server S calculates the weight Enc(w_i) following a DSE algorithm for each user u_i using HE operations based on the received encrypted dataset Enc($\mathcal{D}_i$), encrypted model Enc($\mathcal{M}_i$) from each user u_i, public global FL model $\mathcal{M}$, and benchmark dataset $\mathcal{D}_s$. Then the server S sends encrypted weight Enc(w_i) to the user u_i and the user decrypts the weight and publishes w_i to the all participants. Note that the user holds the private key, and the decryption of weight is on the user-side. Thus, the user may perform fraudulent decryption of the weight. ZKP is adopted here to guarantee the correctness.

However, the user may still upload fraudulent masked weighted model $w_i\mathcal{M}_i + R_i$ to the server for its benefit. Addressing such issue is not trivial as the solution may involve complicated incentive mechanisms. Luckily, since the real encrypted user's model Enc($\mathcal{M}_i$) and encrypted mask Enc(R_i) have already been uploaded to the server at the very beginning, and the verified (real) weight of each user is public, the ZKP technique can be adopted here to guarantee the correctness. In specific, the server can obtain the real masked weighted model in ciphertext by computing $w_i\mathrm{Enc}(\mathcal{M}_i) + \mathrm{Enc}(R_i)$ using HE operations, and then asks the user to upload its weighted model $w_i\mathcal{M}_i + R_i$ for aggregation. In this case, by having both the plaintext and ciphertext of the masked weighted model, the server can require each user to prove the plaintext knowledge of its masked weighted model. Note that as the scheme requires users to upload the encrypted model $\mathrm{Enc}(\mathcal{M}_i)$ and encrypted mask matrix $\mathrm{Enc}(R_i)$ at the very beginning of each round of FL, the users will not deviate from the protocol as they do not know how to design $\mathcal{M}_i$ or R_i to manipulate their weights for benefits without knowing the information of their weights and locally trained models in the current round [8]. After that, the server and users can jointly aggregate the verified weighted models using standard secure aggregation protocol.

4 Example Application

This section describes an example application of our proposed scheme in detail. Specifically, we use Paillier [14] as the HE scheme and devise the Paillier based ZKP of plaintext knowledge technique for verification. We adopt the state-of-the-art secure aggregation scheme [6,7] (see Sect. 2.2) for efficiency and dropout-resilience. The DSE algorithm is from [4] that allows the server to quantify the label quality of each user's dataset (See Sect. 2.1). Besides, we adapt the DSE algorithm to dropped users to keep the consistency of dropout-resilient secure aggregation protocol.

Recall that the server S maintains a benchmark dataset $\mathcal{D}_s = \{(x_s, y_s) | s = 1, 2, \ldots, n_s\}$ and a global FL model $\mathcal{M}$, and the user u_i holds a local dataset $\mathcal{D}_i = \{(x_u, y_u) | u = 1, 2, \ldots, n_i\}$ and a locally trained model $\mathcal{M}_i$ as well as a Paillier key pair (sk_i, pk_i). Before proceeding to the data significance evaluation (computing the weight), each user u_i uses its pk_i to encrypt its dataset and locally trained model as $\mathsf{Enc}(\mathcal{D}_i) = \{(\mathsf{Enc}(x_u), \mathsf{Enc}(y_u)) |\ u = 1, 2, \ldots, n_i\}$ and $\mathsf{Enc}(\mathcal{M}_i)$ respectively, and uploads them to the server. After that, the server is able to compute the weight using HE operations.

4.1 Data Significance Evaluation

The privacy-preserving data significance evaluation works as follows (see Sect. 2.1 for the version over plaintext). First, the server computes and sends the mutual cross entropy E_i in ciphertext $\mathsf{Enc}(E_i)$ to the user i as:

$$\mathsf{Enc}(E_i) = \mathsf{Enc}(LS_i) \boxplus \mathsf{Enc}(LL_i) \tag{1}$$

$$\mathsf{Enc}(LS_i) = -\sum\nolimits_{(x_s, y_s) \in \mathcal{D}_s} y_s \boxtimes \log P\left(y \mid x_s; \mathsf{Enc}(\mathcal{M}_i)\right) \tag{2}$$

$$\mathsf{Enc}(LL_i) = -\sum\nolimits_{(x_u, y_u) \in \mathcal{D}_i} \mathsf{Enc}(y_u) \boxtimes \log P\left(y \mid \mathsf{Enc}(x_u); \mathcal{M}_s\right) \tag{3}$$

Note that the computation of LS_i and LL_i involves calculating non-linear functions which cannot be directly evaluated using HE operations. A common method to address this issue is to replace these non-linear functions with polynomials. For instance, for logistic regression model $l = \sigma(\boldsymbol{\theta} \cdot \boldsymbol{x})$ where θ is the hyper-parameter and σ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, in terms of a good tradeoff between efficiency and accuracy, the sigmoid function can be approximated by a cubic polynomial $\sigma(x) = s_0 + s_1 x + s_2 x^2 + s_3 x^3$, and thus the y in Eq. 2 and 3 can be calculated by $y = \sigma(l) = s_0 + s_1 l + s_2 l^2 + s_3 l^3$. Since in our scheme, either one of $\boldsymbol{\theta}$ and $\boldsymbol{x}$ is in ciphertext, the server can only obtain $\mathsf{Enc}(l)$, which means that the n-th power of l cannot be calculated using Paillier as it does not support multiplication of ciphertext. Therefore, one more round of communication is needed to compute y. Specifically, the server chooses a random value $r \in 2^{\kappa}$ to mask l as $z = l + r$ where κ is the security parameter, and sends $\mathsf{Enc}(z) = \mathsf{Enc}(l) \boxplus \mathsf{Enc}(r)$ to the user. Then the user decrypts $\mathsf{Enc}(z)$, computes $\mathsf{Enc}(z^2)$ and $\mathsf{Enc}(\sigma(z))$ from z and sends $(\mathsf{Enc}(z^2), \mathsf{Enc}(\sigma(z)))$ to the server. Since $y = \sigma(l) = \sigma(z - r) = \sigma(z) - \sigma(r) + \left(s_0 + 3s_3 r^3\right) - 3s_3 r z^2 - \left(2s_2 r - 3s_3 r^2\right) l$, the $\mathsf{Enc}(y)$ can be calculated by

$$\begin{aligned} \mathsf{Enc}(y) = & \mathsf{Enc}(\sigma(z)) \boxplus \mathsf{Enc}(-r) \boxplus \mathsf{Enc}(s_0 + 3s_3 r^3) \\ & \boxplus ((-3s_3 r)\mathsf{Enc}(z^2)) \boxplus ((-2s_2 r + 3s_3 r^2)\mathsf{Enc}(l)). \end{aligned} \tag{4}$$

Note that since $l = \boldsymbol{\theta} \cdot \boldsymbol{x}$ is masked by r, the user has no information of how to design fraudulent $(\mathsf{Enc}(z^2), \mathsf{Enc}(\sigma(z)))$ for its benefit. In addition, there is a difference between the computation of $\mathsf{Enc}(LS_i)$ and $\mathsf{Enc}(LL_i)$ as the former involves multiplication of plaintext while the latter involves multiplication of ciphertext. This means that Eq. 2 can be directly evaluated using Paillier and Eq. 3 needs one more round for computation. Specifically, let $\mathsf{Enc}(h) = \log P\left(y \mid \mathsf{Enc}(x_u); \mathcal{M}^s\right)$ in Eq. 3, the server first sends $\mathsf{Enc}(h)$ to the user for decryption. To protect h which may leak information of x against the server, the user decrypts $\mathsf{Enc}(h)$ and masks h using a randomly chosen value r then sends $h + r$ to the server. Following that, the server computes $\mathsf{Enc}(y_u h + y_u r) = \mathsf{Enc}(y_u) \boxtimes (h + r)$. Assume that $\mathsf{Enc}(y_u r)$ has been uploaded to the server, the server is able to compute $\mathsf{Enc}(y_u h) = \mathsf{Enc}(y_u h + y_u r) \boxplus \mathsf{Enc}(-y_u r)$. Note that r is secure even if the server holds $\mathsf{Enc}(y_u)$ and $\mathsf{Enc}(y_u r)$. Such security is provided by the IND-CPA property of Paillier. In addition, for computing the cross-entropy after sigmod function, we can further simplify the polynomial replacement of their combined function using the similar above mentioned method.

After that, each user i decrypts $\mathsf{Enc}(E_i)$ and publishes E_i. Finally, all participants can compute the credibility C_i and weight w_i for each user i following:

$$w_i = \frac{n_i C_i}{\sum_{k=1}^{K} n_k C_i} = \frac{n_i e^{\alpha RE_i}}{\sum_k n_k e^{\alpha RE_k}}, \text{ such that} \sum_i w_i = 1 \tag{5}$$

where $C_i = \frac{e^{\alpha RE_i}}{\sum_k e^{\alpha RE_k}}$, $RE_i = \frac{1}{E_i}$. Here Eq. 5 is the adapted version of that in [4] in order to handle dropped users by simply adjusting the denominator of w_i in Eq. 5 on server-side during subsequent weighted aggregation (see Sect. 4.2).

As discussed in Sect. 3, since the user i may deviate from the protocol by publishing fraudulent decryption of E_i, a verification scheme is needed to guarantee the correct decryption of E_i on the user-side. To address this issue, we construct a variant of the ZKPoPK of Zero algorithm given in Sect. 2.3 to enable a user u_i with a key pair (pk_i, sk_i) to convince the server S the fact that a message m is the correct decryption of ciphertext c. To do so, given the message m, the server can compute $c' = c - \mathsf{Enc}(m)$ and then requires the user to prove the knowledge of zero of the ciphertext c' following Algorithm 1. The detail of this algorithm is given in Algorithm 2.

Assume that the server receives a message E'_i which is the decryption of $\mathsf{Enc}(E_i)$ from the user u_i. Following Algorithm 2, the server S and the user u_i get $\beta \leftarrow \mathsf{PPoPK}(\mathsf{Enc}(E_i),\ E'_i,\ pk_i, sk_i)$. The server accepts E'_i as the plaintext of $\mathsf{Enc}(E_i)$ if and only if $\beta = 1$, which means the user u_i honestly decrypts $\mathsf{Enc}(E_i)$. Otherwise, the server refuses the message E'_i and removes the user u_i from the user set for aggregation. Similar approach can be adopted to guarantee the correct decryption of h.

Algorithm 2. Paillier-based PoPK $\beta \leftarrow \mathsf{PPoPK}(c, m, pk_i, sk_i)$

Public input: $c, m, pk_i = (n, g)$.
Private input for the user u_i**:** $sk_i = (p, q)$
Output: β

```
1: The server randomly chooses $r_s \in \mathbb{Z}_n^*$, and computes $c' = c - \mathsf{HE.Enc}(pk_i, m, r_s)(\bmod n^2)$. The server sends $c'$ to the user.
2: The user computes $r' = c'^d \bmod n$, where $d = n^{-1} \bmod \varphi(n)$.
3: The server and the user jointly follow Algorithm 1 to get $\beta \leftarrow \mathsf{PZKPoPKoZ}(n, c', pk_i, sk_i, r')$.
```

The correctness is proved as follows. If the message m provided by the user is the decryption of c, $c' = c - \mathsf{HE.Enc}(pk_i, m, r_s)(\bmod n^2)$ should be the ciphertext of zero with respect to r' that $c' \bmod n^2 = (c')^{n \cdot d} \bmod n^2 = (r')^n \bmod n^2 = \mathsf{HE.Enc}(pk_i, 0, r')$. Therefore, the output β of the algorithm $\mathsf{PZKPoPKoZ}$ in step 3 is 1. Similarly, if m is not the decryption of c, the algorithm $\mathsf{PZKPoPKoZ}$ outputs $\beta = 0$.

Theorem 1 (Security against malicious users). *The Paillier-based Proof of Plaintext Knowledge is secure against malicious users. If there exists a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ with Paillier key pair (sk, pk) providing a fraudulent plaintext m' as the decryption of c that $\mathsf{HE.Dec}(sk, c) \neq m'$. The probability of the server accepting the fraudulent decryption $\mathsf{Pr}[1 \leftarrow \mathsf{PPoPK}_{\mathcal{A}}(c, m', pk, sk)] \leqslant 2^{-k/2}$ where k is the bit length of n as the Paillier public key.*

Proof. Assume that a PPT adversary $\mathcal{A}$ with key pair (sk, pk) provides a fraudulent decryption m' of a ciphertext c, while $\mathsf{HE.Dec}\ (sk, c) = m$ and $m \neq m'$, the server accepts m' only if $1 \leftarrow \mathsf{PPoPK}(c,\ m', pk, sk)$. Given the Algorithm 2, the server randomly chooses $r_s \in \mathbb{Z}_n^*$ to compute $c' = c - \mathsf{HE.Enc}(pk_i, m', r_s)(\bmod\ n^2) = \mathsf{Enc}(m - m')$ and sends c' to the adversary

$\mathcal{A}$. To make the server accept m', the adversary $\mathcal{A}$ has to prove that c' is a ciphertext of zero. Therefore, the security of PPoPK follows from the security of PZKPoPKoZ such that the probability of the adversary make the server accept the fraudulent decryption m' is less than $2^{-k/2}$ where k is the bit length of n as the Paillier public key.

4.2 Weighted Aggregation

After obtained the verified weights, we can proceed to the weighted aggregation where the server can securely aggregate user's locally trained models to update the global FL model $\mathcal{M}$ according to their corresponding weights w, i.e. $\mathcal{M} = \sum_{u_i \in \mathcal{U}'} w_i \mathcal{M}_i$ such that $\sum_{u_i \in \mathcal{U}'} w_i = 1$, where w_i and $\mathcal{M}_i$ are the weight and local model of user u_i, and $\mathcal{U}'$ is the selected user set in the current round of FL.

Let the numerator of w_i in Eq. 5 be $\omega_i = n_i e^{\alpha RE_i}$, we can rewrite Eq. 5:

$$\mathcal{M} = \frac{1}{\sum_{u_j \in \mathcal{U}'} n_j e^{\alpha RE_j}} \sum_{u_i \in \mathcal{U}'} \omega_i \mathcal{M}_i \tag{6}$$

Therefore, the server can deal with dropped users by adjusting $\mathcal{U}'$ and aggregates the models from alive users.

To prevent users from uploading fraudulent weighted models to the server, as we discussed in Sect. 3, the server first computes

$$\mathrm{Enc}(y_i) = \omega_i \boxtimes \mathsf{Enc}(\mathcal{M}_i) \boxplus \mathsf{Enc}(R_i) \tag{7}$$

Then the server can verify if the user uploads its real masked weighted model relying on the Algorithm 2 with zero knowledge by checking the equality between y_i and uploaded masked weighted model y_i'.

Note that the $\mathsf{Enc}(R_i)$ should be uploaded at the very beginning of each FL round according to the analysis in Sect. 3 to prevent the user from manipulating R_i for their benefits. Besides, since verified weight ω_i is a public value and $\mathsf{Enc}(\mathcal{M}_i)$ has been uploaded during data significance evaluation, the server can calculate y_i' using Pallier HE operations. The detailed description of this example of our proposed scheme for one FL round is given in Protocol 4.1. Note that the **Setup** step is only performed once at the very beginning of federated learning. Step 0 to step 4 are performed in every FL round.

4.3 Discussion on Privacy and Security

First, we give the discussion from the angle of privacy protection. Since both the user's dataset $\mathcal{D}_i$ and locally trained model $\mathcal{M}_i$ are encrypted in step Setup and step 1, then uploaded to the server, the server learns nothing from the ciphertext. After that, during computing the weight in step 1, HE operations are applied, and thus additions between ciphertexts do not leak any information. For computing a polynomial $f(l)$ which consists of multiplication between ciphertexts where the server keeps $\mathrm{Enc}(l)$, a trick is adopted as described in Sect. 2.1. The main idea is to use a random r to mask l to enable the computation $f(l + r)$ over plaintext, then deduct the redundant terms of $f(l + r)$ to obtain the encryption of $f(l)$ using HE. The security of the trick is based on that the user does not know the value of r. The privacy of the public masked weighted model $w_i\mathcal{M}_i + R_i$ in step 3 and step 4 is protected by the mask R_i randomly chosen by the user u_i. Therefore, there is no information leakage of both user's dataset and model during the execution of our proposed protocol.

Next, we discuss from the angle of security against fraudulent messages. Since the server has the encryption of the real user's model y_i, real mutual cross-entropy E_i, and real mask R_i after step setup, step 0, and step 1, while zero-knowledge proof is adopted in step 2 to ensure the correct decryption of E_i, both the server and user can compute real w_i, hence the encryption of real masked weighted model $w_i\mathcal{M}_i + R_i$, which can be used for the zero-knowledge proof in step 3. Therefore, our scheme guarantees security against fraudulent messages regarding both the users' datesets, models, and the computed weights.

5 Performance Analysis

System Setting. Our prototype is tested over two Linux workstations with an Intel Xeon E5-2603v4 CPU (1.70 GHz) and 64 GB of RAM, running CentOS 7 in the same region. The average latency of the network is 0.207 ms, and the average bandwidth is 1.25 GB/s. The central server runs in multi-thread on one workstation, while the users are running in parallel in the other workstation. In our experiments, the ring size N for Paillier is set to be a 1024-bit integer, and the security parameter κ is set to be 80. We use AES-GCM with 128-bit keys for authenticated channels, and adopt SecAgg+ scheme [7] for dropout-resilient aggregation. The performance is evaluated on several datasets from UCI repository from which each user randomly chooses samples of the same size to construct its local dataset. The details are given in Table 1.

As seen in Fig. 2, the accuracy of the global FL model before and after using our scheme shows a consistent tolerance of fraudulent messages. We can observe that sending fraudulent weights (assign $E_i = 1$) results in much fewer effects than sending fraudulent models. This indicates the difficulty of detecting fraudulent weights and the significant adverse impact of sending fraudulent models, hence the necessity of our proposed scheme. In addition, Fig. 3 shows the convergence performance when fraudulent messages exist in the FL system with and without our scheme. We should note that the tolerance of fraudulent messages on the performance of the FL model varies from datasets and models.

We also evaluate the scalability of our proposed scheme with respect to the number of parties and dropout rate, compared to the model aggregation counterpart of SecAgg [6] and the improved version SecAgg+ [7]. Note that we treat the users who do not pass PoKE and PoKM as dropout users as demonstrated in Protocol 4.1. In specific, we consider the "worst-case" dropout scenario where the users drop out during PoKE in the first phase as the server has already completed the calculation of the weights, or PoKM in the second phase as the users have already uploaded their weighted models. In Table 2, we can observe

Protocol: Secure Weighted Aggregation

Private inputs: Each user u_i has a dataset $\mathcal{D}_i$, a locally trained model $\mathcal{M}_i$, and a Paillier key pair (sk_i, pk_i). Sever has a benchmark dataset $\mathcal{D}_s$.
Public inputs: Each user u_i's Paillier public key pk_i. The global FL model $\mathcal{M}$, the selected user set $\mathcal{U}' \subseteq \mathcal{U}$, and a threshold t for Shamir secret sharing scheme.
Outputs: The weighted aggregation of all local models from the alive users $\sum_i w_i \mathcal{M}_i$.

- **Setup - Uploading Encrypted Dataset (performed only once):**
  User u_i:
  1. Uses pk_i to encrypt and send $\mathsf{Enc}(\mathcal{D}_i) = \{(\mathsf{Enc}(x_u), \mathsf{Enc}(y_u))\}$ to the server.

  Server:
  1. Receives $\mathsf{Enc}(\mathcal{D}_i)$ from user $u_i \in \mathcal{U}$.
- **Step init - Initialization (Init) :**
  User u_i:
  1. Computes the matrix R_i according to π_{MG}, such that $(\{R_i\}_{u_i \in \mathcal{U}_1}, \mathcal{U}_1) \leftarrow \pi_{MG}(t, \mathcal{U})$, where the output $\mathcal{U}_1$ is the set of alive users.
  2. Uses the local dataset $\mathcal{D}_i$ to train the local model $\mathcal{M}_i$.
  3. Encrypts R_i, $\mathcal{M}_i$, and then sends $\{\mathsf{Enc}(R_i), \mathsf{Enc}(\mathcal{M}_i)\}$ to the server.

  Server:
  1. Collects $\{\mathsf{Enc}(R_i), \mathsf{Enc}(\mathcal{M}_i)\}$ from the set of alive users denoted as $\mathcal{U}_2 \subseteq \mathcal{U}_1$.
- **Step 1 - Computing Mutual Cross-entropy (CompE) :**
  Server:
  1. For each user $u_i \in \mathcal{U}_2$, computes the mutual cross-entropy $\mathsf{Enc}(E_i)$ of the user u_i based on Eq. 1, 2 and 3, and then sents it to user u_i.
- **Step 2 - Proof of Knowledge of Mutual Cross-entropy (PoKE) :**
  1. $\underline{\text{The user } u_i \text{ decrypts } \mathsf{Enc}(E_i) \text{ to get } E_i' \text{ and sends it to the server.}}$
  2. Computes $\beta_E \leftarrow \mathsf{PPoPK}(\mathsf{Enc}(E_i), E_i', pk_i, sk_i)$ following Alg. 2.
  3. If $\beta_E = 0$, the server removes u_i from $\mathcal{U}_2$. Denote the rest of alive users as $\mathcal{U}_3$. Else both the user u_i and the server compute ω_i in Eq. 6.
- **Step 3 - Proof of Knowledge of Masked Weighted Model (PoKM) :**
  User u_i:
  1. $\underline{\text{Uploads } y_i' = w_i \mathcal{M}_i + R_i \text{ to the server.}}$

  Server:
  1. Receives y_i' from the set of current alive users denoted as $\mathcal{U}_4$. For each user $u_i \in \mathcal{U}_4$, computes $\mathrm{Enc}(y_i) = \omega_i \boxtimes \mathsf{Enc}(\mathcal{M}_i) \boxplus \mathsf{Enc}(R_i)$ according to Eq. 7.
  2. Computes $\beta_M \leftarrow \mathsf{PPoPK}(\mathrm{Enc}(y_i), y_i', pk_i, sk_i)$ following Alg. 2
  3. If $\beta_M = 0$, the server removes u_i from $\mathcal{U}_4$, denote the rest of alive users as $\mathcal{U}_5$.
  4. Computes the aggregation of masked weighted models such that $y = \sum_{u_i \in \mathcal{U}_5} y_{u_i}$ following π_{MMA} protocol.
- **Step 4 - Weighted Aggregation (WAgg) :**
  Server:
  1. Publishes the alive user set $\mathcal{U}_5$, and cooperates with clients $u_i \in \mathcal{U}_5$ to jointly compute and output the final weighted aggregation $\mathcal{M}$ according to the protocol π_{MAR}, such that $\mathcal{M} \leftarrow \pi_{MAR}(y, \mathcal{U}_1, \mathcal{U}_5, t)$.

Protocol 4.1. Detailed description of proposed privacy-enhanced FL scheme. The participants may deviate from the protocol in the underlined operations.

that dropout users cause a significant increasement in the total run time because of the high cost of dealing with dropout users in the underlying secure aggregation scheme. As shown in Fig. 4, the experimental results show that our scheme provides an additional security guarantee against fraudulent messages with affordable overheads compared to the previous works. In particular, our scheme achieves around 1.2 times in run time and 1.3 times in communication cost upon the state-of-the-art secure aggregation scheme SecAgg+, which indicates the practicality of our scheme.

Table 1. Datasets used in our experiments. The number of clients is fixed to 100.

Dataset	#Attributes	#total samples	#Training samples per client	#Sample of benchmark dataset
ADULT [17]	14	48842	200	15060
CREDIT [18]	24	30000	240	10000
BK [19]	17	45211	350	9042
RFID-AC [20]	6	75128	400	10497

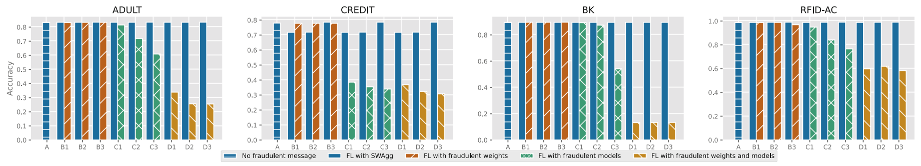
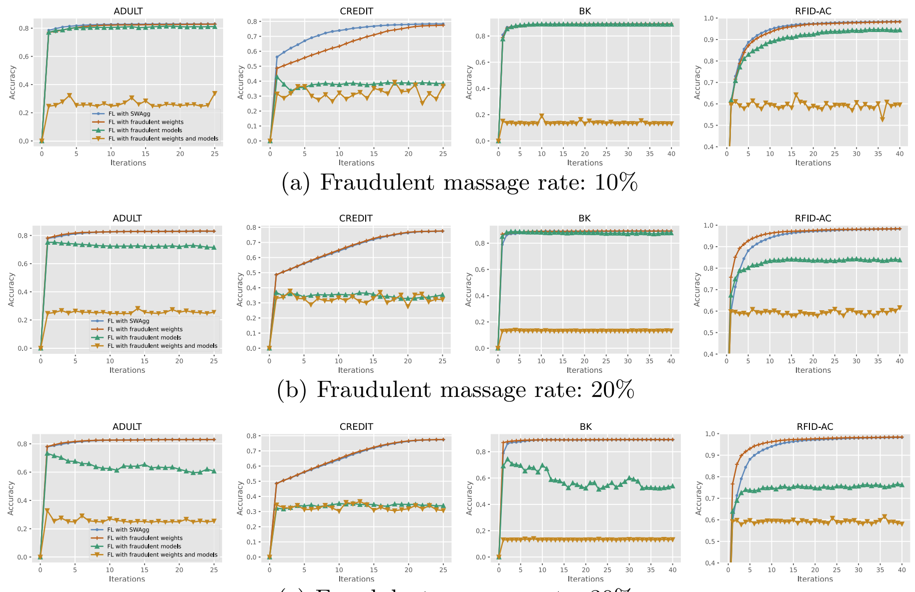


Fig. 2. FL performance over different fraudulent messages. Note that the meaning of x-axis labels are A: no fraudulent message; B1, B2, B3: the fraction of clients sending fraudulent weights 10%, 20%, 30% (similar setting of SecAgg [6]); C1, C2, C3: the fraction of clients sending fraudulent models 10%, 20%, 30%; D1, D2, D3: the fraction of clients sending both fraudulent weights and models 10%, 20%, 30%.

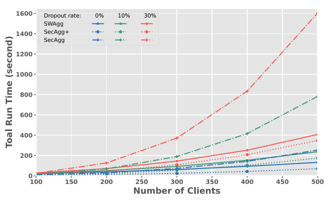


(a) Fraudulent massage rate: 10%

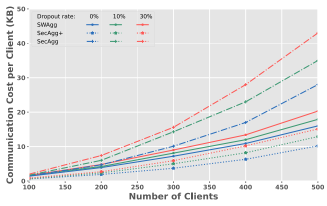
(b) Fraudulent massage rate: 20%

(c) Fraudulent massage rate: 30%

Fig. 3. The convergence performance of FL with different fraudulent message rate. The number of clients is fixed to 100.



(a) Run time



(b) Communication cost

Fig. 4. The performance of our proposed scheme SWAgg compared with SecAgg [6] and SecAgg+ [7] with different dropout rate.

Table 2. The run time (s)/communication costs (MB) of the user (each) and the server (with all users) for each step of our proposed scheme in one FL round. The number of users is set to be 500. R_1 and R_2 are the fractions of dropout users in PoKE and PoKM, respectively.

	R_1	R_2	Init	ComE	PoKE	PoKM	WAgg	Total
User	0	0	0.93/0.48	5.65/1.02	0.01/0.01	0.02/0.01	0.04/0.02	6.65/1.54
Server	0	0	1.22/240	10.97/512	0.04/3	0.13/6	67.31/10	79.67/771
Server	10%	0	1.24/240	10.92/513	0.04/3	0.12/6	173.27/11	185.59/773
Server	0	10%	1.24/240	10.92/513	0.04/4	0.13/5	172.80/12	185.13/774
Server	5%	5%	1.24/240	10.95/513	0.04/3	0.12/5	172.16/13	184.51/774
Server	30%	0	1.23/241	10.92/512	0.04/3	0.10/4	351.78/14	364.07/774
Server	0	30%	1.24/240	10.95/512	0.04/4	0.12/4	354.40/15	366.75/775
Server	15%	15%	1.23/240	10.92/513	0.04/3	0.11/3	349.07/15	361.37/775

6 Conclusion

In this paper, we proposed a privacy-enhanced FL scheme for supporting secure weighted aggregation. Our scheme is able to deal with both data disparity, data privacy, and dishonest participants (who send fraudulent messages to manipulate the computed weights) in FL systems. Experimental results show that our scheme is practical and secure. Compared to existing FL approaches, our scheme achieves secure weighted aggregation with an additional security guarantee against fraudulent messages with affordable runtime overheads and communication costs.

References

1. Zhao, Y., et al.: Local differential privacy based federated learning for Internet of Things. IEEE Internet Things J. **8**(11), 8836–8853 (2021)

2. Yang, H., Zhao, J., Xiong, Z., Lam, K.Y., Sun, S., Xiao, L.: Privacy-preserving federated learning for UAV-enabled networks: learning-based joint scheduling and resource management. IEEE J. Sel. Areas Commun. **39**(10), 3144–3159 (2021)
3. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, pp. 1273–1282. PMLR (2017)
4. Chen, Y., Yang, X., Qin, X., Yu, H., Chen, B., Shen, Z.: Focus: dealing with label quality disparity in federated learning. In: International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with IJCAI 2020 (2020)
5. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: Advances in Neural Information Processing Systems, Vancouver, BC, Canada, pp. 14774–14784. NeurIPS (2019)
6. Bonawitz, K., et al.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, pp. 1175–1191. ACM (2017)
7. Bell, J.H., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly) logarithmic overhead. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, pp. 1253–1269. ACM (2020)
8. Blanchard, P., El Mhamdi, E.M., Guerraoui, R., Stainer, J.: Machine learning with adversaries: Byzantine tolerant gradient descent. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, pp. 118–128. NIPS (2017)
9. McMahan, H.B., Ramage, D., Talwar, K., Zhang, L.: Learning differentially private recurrent language models. In: International Conference on Learning Representations, BC, Canada (2018)
10. Rastogi, V., Nath, S.: Differentially private aggregation of distributed time-series with transformation and encryption. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, Indianapolis, Indiana, USA, pp. 735–746. ACM (2010)
11. So, J., Güler, B., Avestimehr, A.S.: Turbo-aggregate: breaking the quadratic aggregation barrier in secure federated learning. IEEE J. Sel. Areas Inf. Theory **2**(1), 479–489 (2021)
12. Mandal, K., Gong, G., Liu, C.: Nike-based fast privacy-preserving highdimensional data aggregation for mobile devices. Technical report, CACR Technical Report, CACR 2018–10, University of Waterloo, Canada (2018)
13. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
14. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
15. Damgård, I., Jurik, M., Nielsen, J.B.: A generalization of Paillier's public-key system with applications to electronic voting. Int. J. Inf. Secur. **9**(6), 371–385 (2010)
16. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science, Las Vegas, Nevada, USA, pp. 136–145. IEEE (2001)
17. Kohavi, R.: Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, USA, pp. 202–207. AAAI (1996)

18. Yeh, I.C., Lien, C.: The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Syst. Appl. **36**(2), 2473–2480 (2009)
19. Moro, S., Cortez, P., Rita, P.: A data-driven approach to predict the success of bank telemarketing. Decis. Support Syst. **62**, 22–31 (2014)
20. Torres, R.L.S., Ranasinghe, D.C., Shi, Q., Sample, A.P.: Sensor enabled wearable RFID technology for mitigating the risk of falls near beds. In: 2013 IEEE International Conference on RFID (RFID), pp. 191–198. IEEE (2013)