Why Not Model Privacy?: An Efficient and Practical Scheme for Federated Learning Model Security

Wang Shuai[1], Renwan Bi[2], Youliang Tian[1(✉)], and Jinbo Xiong[2]

[1] The State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China
youliangtian@163.com

[2] The Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China
jbxiong@fjnu.edu.cn

Abstract. Privacy-Preserving Federated Learning (PPFL), a new paradigm for secure and efficient Federated Learning, has the advantage of security aggregation without losing accuracy. However, PPFL focuses on protecting the privacy of private data, but the privacy of models is equally important. Model sharing among institutions increases the risk of model privacy leakage and economic loss when federated learning is applied to real-world scenarios. To address the above issues, we design a model privacy-preserving scheme based on fully homomorphic encryption, PriM. Specifically, we adopt the CKKS fully homomorphic encryption scheme to guarantee the security of the model, which supports both floating-point and vector encryption and avoids the significant overhead of single-value encryption in the homomorphic encryption-based PPFL scheme. Then, we maintain the confidentiality of the model computation process without revealing the model information to the participating entities. Finally, experiments show that our scheme achieves equal efficiency with the baseline algorithm FedAvg but is more realistic.

Keywords: privacy-preserving federated learning · homomorphic encryption · model security

1 Introduction

Federated learning (FL) has become a new paradigm for large-scale distributed machine learning, collaborating with numerous end devices through aggregated servers to complete model training and update tasks while preserving client data locally [1]. In FL, because of its distributed architecture and collaborative approach, there are many tricks (e.g., inference attacks [2], inversion attacks [3], backdoor attacks [4] and poisoning attacks [5]) that make the client data and aggregated intermediate values risky for privacy leakage. Hence, Privacy-Preserving Federated Learning (PPFL) has attracted widespread attention in

J. Wang (Ed.): MobiMedia 2023, LNICST 576, pp. 400–410, 2025.
https://doi.org/10.1007/978-3-031-60347-1_33

both academia and industry [6–8] as a scheme for securing data aggregation using encryption [9–11], differential privacy [12,13] or secure multi-party computing [14,15] techniques.

However, PPFL also suffers many practical challenges in real-world applications [16,17]. Existing research focuses more on the privacy preservation of clients' raw data, which is the original intention of traditional FL. Still, the security of the model is equally important. For example, when a company (task publisher) adopts a cloud server (aggregation server) machine learning model to support decision-making, the aggregation server randomly selects some mobile devices (clients) to participate in the training. Some of these clients have active or passive malicious operations that cause the model to be compromised or apologized; even worse, it can lead to severe business losses for the task publisher [18,19]. Additionally, the models often contain sensitive information of other clients, which may lead to personal privacy leakage if attackers access the models. In general, it is of practical significance and urgent to adopt privacy-preserving techniques to protect machine learning models [20].

Homomorphic encryption is widely utilized in PPFL, privacy-preserving data mining, and intelligent contracts because of its advantage of computing ciphertexts without exposing plaintext data [21]. In previous application scenarios, however, research has mainly utilized homomorphic encryption to secure raw data, and in contrast to existing research, we first apply homomorphic techniques to model privacy protection. Specifically, to address the high computational complexity of traditional homomorphic encryption that is difficult to apply in large-scale machine learning algorithms, we use a typical fully homomorphic encryption scheme CKKS [22] combined with FL to enhance the computational efficiency of data encryption. In this paper, this is the first solution that applies homomorphic encryption to protect the FL model. First, we propose an efficient and practical security framework, PriM, to ensure the privacy availability of the FL model. Second, the CKKS fully homomorphic encryption scheme is applied to model encryption to support vector encryption, which speeds up the federated learning execution process. Finally, preliminary experiments show the efficiency and practicality of PriM compared with existing schemes.

2 Related Work

In this paper, we focus our research around existing schemes that combine homomorphic encryption with FL, and focus on related work in FL on the following related work. Researchers commonly employ homomorphic encryption techniques to protect the privacy of raw client data in FL because of its significant property of computational equivalence in the plaintext/ciphertext domain. Phong *et al.*. [23] proposed a privacy-preserving deep learning framework based on additive homomorphic encryption. Where the initialized global model is generated by a client trained on its local dataset and encrypted to the server, the rest of the clients download the encrypted weight parameters and decrypt them to update the model on their local dataset, as well as send the updated gradients encrypted to the server, which directly aggregates them with the ciphertext

gradients to get a fresh global model, repeating the process until the model converges or achieves the maximum iteration. Xu *et al.*. [24] designed a CryptoNN framework, which ensures the privacy of user data through function encryption techniques. In this scheme the authors only encrypt between the input layer and the first hidden layer of the feed-forward step, as well as between the output layer and the last hidden layer of the backward-propagation process, ignoring the neural network details in the middle of the hidden layers. Zhao *et al.*. [25] seamlessly integrate FL into mobile crowdsensing(MCS) and design a privacy-preserving MCS system CROWDFL based on a threshold Paiilier cryptosystem. In this framework, the requester divides the private key into two parts, and the sensing platform (server) and participants (clients) hold one private key respectively. Similarly, clients collect data and train a model on them, encrypt the model using the client's private key and upload it to the server, where the server aggregates and partially decrypts the encrypted model, and then returns the partially decrypted aggregated model to clients. Finally, clients obtains the aggregated model after full decryption and update with the plaintext aggregated model.

Surprisingly, the global models in the above solutions are finally decrypted at the client to get the plaintext models, which means that the global models are publicly exposed to the computing procedures of the system. Existing researches tend to make honest and curious or malicious assumptions about clients, leading to model privacy leakage. Likewise, in specific real-world scenarios where participants are willing to pay lower costs to intercept information about the model, model leakage inevitably results in significant economic damages. However, our proposed PriM framework emphasizes more on model security with efficient fully homomorphic encryption to encrypt all model parameters and maintain ciphertext computation during the computation process, without disclosing the model to participants, much less leaking model parameters.

3 Preliminaries

In this section we briefly introduce the fully homomorphic scheme.

3.1 CKKS Fully Homomorphic Encryption

CKKS [22] approximate arithmetic homomorphic encryption scheme, as a typical full homomorphic encryption scheme, can support approximate computation of floats and vectors. CKKS has unique coding and decoding techniques and rescaling mechanisms. We describe briefly the CKKS algorithm composition in the following:

- **KeyGen**$(1^\lambda) \rightarrow (sk, pk, evk)$.
  - Given the security parameter λ, the base of the scaling approximation calculation $p > 0$, a modulus q_0 and $0 < l < L$, the ciphertext is a vector in $\mathcal{R}_{q_l}^k$ of integer k with size l, where $q_l = p^l \times q_0$. Select a power-of-two $M = M(\lambda, q_l)$, the integer $h = h(\lambda, q_l)$ and $P = P(\lambda, q_l)$, a real number $\sigma = \sigma(\lambda, q_l)$.

  - Sample $s \leftarrow \mathcal{H}(h), a \leftarrow \mathcal{R}_{q_l}$ and $e \leftarrow \mathcal{G}(\sigma^2)$. $\mathcal{H}(h)$ is the set of signed binary vectors in $0, \pm 1^N$ whose Hamming weight is h, $\mathcal{G}(\sigma^2)$ is a vector in $\mathbb{Z}^N$ that follows a discrete Gaussian distribution of variance σ^2. Given the secret key as $sk \leftarrow (1, s)$ and the public key as $pk \leftarrow (b, a) \in \mathcal{R}_{q_l}^2$ where $b \leftarrow -as + e(\mod q_l)$.
  - Sample $a' \leftarrow \mathcal{R}_{P \cdot q_l}$ and $e' \leftarrow \mathcal{G}(\sigma^2)$. Set the evaluation key as $evk \leftarrow (b', a') \in \mathcal{R}_{P \cdot q_l}^2$ where $b' \leftarrow -a's + e' + Ps^2(\mod P \cdot q_l)$.
- $\mathbf{Ecd}(\mathbf{z}, s) \rightarrow m$. Let T be a subgroup of $\mathbb{Z}_M^*$ satisfying $\mathbb{Z}_M^*/T = \pm 1$. For a $(N/2)$-dimensional vector $\mathbf{z} = (z_j)_{j \in T} \in \mathbb{Z}[i]^{N/2}$ of Gaussian integers, compute the vector $\lfloor s \cdot \pi^{-1}(\mathbf{z}) \rceil_{\sigma(\mathcal{R})}$, return the encoded value $m = \sigma^{-1} \circ \pi^{-1}(s \cdot \mathbf{z})$.
- $\mathbf{Dcd}(m, s) \rightarrow \mathbf{z}$. For an input polynomial $m(X) \in \mathcal{R}$, compute the corresponding vector $\pi \circ \sigma(m)$,return the closest approximation vector of Gaussian integers $\mathbf{z} = (z_j)_{j \in T} \in \mathbb{Z}[i]^{N/2}$ after scaling, i.e., $z_j = \lfloor s^{-1} \cdot m(\zeta_M^j) \rceil, j \in T$.
- $\mathbf{Enc}(pk, m) \rightarrow \mathbf{c}$. For a real number $0 \leq \rho \leq 1$, the distribution $\mathcal{Z}(\rho)$ selects each entry in the vector from $0, \pm 1^N$, with probability $\rho/2$ for each of -1 and $+1$, and probability being zero $1 - \rho$. Sample $v \leftarrow \mathcal{Z}(0.5)$ and $e_0, e_1 \leftarrow \mathcal{G}(\sigma^2)$. Output the ciphertext $\mathbf{c} = v \cdot pk + (m + e_0, e_1)(\mod q_l)$.
- $\mathbf{Dec}(sk, \mathbf{c}) \rightarrow m$. For $\mathbf{c} = (b, a)$, output the plaintext $m = b + a \cdot s(\mod q_l)$.
- $\mathbf{Add}(\mathbf{c}_1, \mathbf{c}_2) \rightarrow \hat{\mathbf{c}}$. For $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{R}_{q_l}^2$, output the added value $\hat{\mathbf{c}} \leftarrow \mathbf{c}_1 + \mathbf{c}_2(\mod q_l)$.
- $\mathbf{Mult}(evk, \mathbf{c}_1, \mathbf{c}_2) \rightarrow \tilde{\mathbf{c}}$. For $\mathbf{c}_1 = (b_1, a_1), \mathbf{c}_2 = (b_2, a_2) \in \mathcal{R}_{q_l}^2$, let $(d_0, d_1, d_2) = (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2)(\mod q_l)$. Output the multiplied value $\tilde{\mathbf{c}} \leftarrow (d_1, d_1) + \lfloor P^{-1} \cdot d_2 \cdot evk \rceil(\mod q_l)$.
- $\mathbf{RS}(\mathbf{c}) \rightarrow \mathbf{c}'$. For $\mathbf{c} \in \mathcal{R}_{q_l}^2$, output the rescaled value $\mathbf{c}' \leftarrow \lfloor \frac{q_{l'}}{q_l} \mathbf{c} \rceil \in \mathcal{R}_{q_{l'}^k}$, where the level $l' < l$.

4 Problem Formulation

In this section, we formalize the system model and threat model, respectively.

4.1 System Model

Specifically, our PriM consists of three types of entities as shown in Fig. 1, *Requester*, *Server*, and *Client*. The roles of each entity are as follows.

- *Requester*: *Requester* may publish tasks that allow the server to select some clients to collaboratively train the model, and *Requester* obtains the final aggregated model from the server. In addition, *Requester* is responsible for generating the security parameters and public/private key pairs in the system.
- *Server*: *Server* takes charge of selecting clients, initializing the global model and encrypting it, and broadcasting the encrypted model to the clients. In addition, *Server* also requires to aggregate the ciphertext models from clients to complete the training of FL. According to the *Server*'s perspective, she knows nothing about the subsequent encrypted models from clients besides the random initialization of the global model at the beginning, while the

model aggregation is also operated under ciphertext. Therefore, the model is not public during the computation of PriM, which achieves the effect of protecting the model security.

- *Client*: *Client* train the ciphertext model on a private dataset and encrypt the trained model to the server. The encrypted model appears to *Client* as messy data, and there is little useful information about the real model accessible to *Client*. Likewise, the model is not public to *Client*.

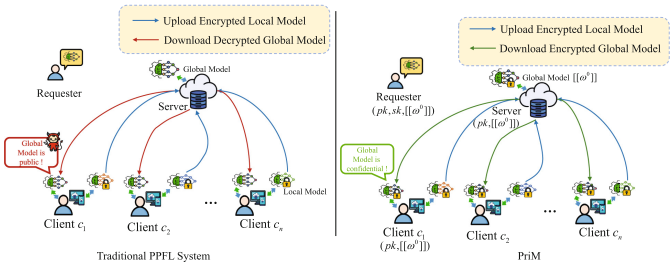


Fig. 1. System model of PriM

4.2 Threat Model

In PriM, *Requester* works as a trusted third party because she, as the task initiator, wants to obtain an optimal model and without revealing information about the model. *Server* and *Client* are "Honest but Curious", which implies that they execute the protocol honestly but will be curious to infer sensitive information. Correspondingly, a valid model is extremely attractive to them, and they would invert the model with intermediate values generated during the interaction of FL, which is poor and causes significant challenges for the security of the model.

5 Proposed Scheme

In this section, we provide the technical overview and illustrate the core ideas of PriM, respectively.

5.1 Technical Overview

Traditional PPFL schemes primarily adopt homomorphic encryption schemes such as Paiilier [23,26] and CKKS [27] to protect gradient privacy. However, Paiilier scheme can only support individual data for encryption, and require element-wise encryption when encrypting the gradient matrix. It brings serious computational overhead and communication overhead, which is not suitable for

large-scale vectors and model with many parameters. Therefore, we use CKKS encryption scheme supporting vector encryption in parallel to encrypt the model that improves the computational efficiency.

In addition, we need to prevent "Honest but Curious" server and clients from potentially inferring sensitive information about the model. Because PriM follows the principles of FL, clients can't upload private data to the server platform nor encrypt the training data, and the model is confidential and non-public at both clients and the server, which makes both model and training data without the threat of privacy leakage. To achieve this, the server receives the encrypted model $[[\omega^0]]$ from the requester and selects a random set of clients $\mathcal{C}$ to participate in the training of FL,then sends $[[\omega^0]]$ to the clients $i, i \in \mathcal{C}$. Client i executes the SGD algorithm to optimize the model in the local dataset $\mathcal{D}_i$ with the computation process keeping ciphertext computation. Meanwhile, the client sends $[[\omega_i^j]]$ to the server after the local training finished, and the server performs aggregation to obtain the new model $[[\omega^{j+1}]] = \sum_{i=1}^{|\mathcal{C}|}[[\omega_i^j]]$, repeating the process until the model converges or reaches the iteration period.

5.2 Construction of PriM

PriM is described in Algorithm1. PriM consists of three phases, initialization, model training, and model aggregation. In the following, we describe each phase in detail.

a) ***Initialization***: *Requester* randomly initializes the global model ω^0 and generates public key pk, private key sk and evaluation key evk. In order to prevent *Sever* and *Client* from learning information about the model, the requester encrypts the model ω_0 and sends the ciphertext model $[[\omega^0]]$ to *Sever*, then broadcasts the public key pk to *Sever* and *Client*.

b) ***Model Training***: *Client* i performs SGD algorithm to update the model on the private dataset $\mathcal{D}_i$ with the received $[[\omega^0]]$, and generates $[[\omega_i^j]]_{k+1}$ and sends it to the server after the local training.

$$[[\omega_i^j]]_{k+1} \leftarrow [[\omega_i^j]]_k - \alpha \nabla L([[\omega_i^j]]_k; \mathcal{D}_i) \tag{1}$$

c) ***Model Aggregation***: After receiving the model $[[\omega_i^j]]$ from clients in $\mathcal{C}$, *Sever* securely performs the aggregation of the ciphertext model because of the homomorphism of the integer vector encryption scheme.

$$[[\omega_i^{j+1}]] \leftarrow \sum_{i=1}^{|\mathcal{C}|} \frac{1}{|\mathcal{C}|}[[\omega_i^j]] \tag{2}$$

Sever returns the completed trained model $[[\omega]]$ to the requester. *Requester* eventually decrypts it using its own private key to obtain a well-formed model.

Algorithm 1: PriM

Input: n clients $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ with local trainning datasets $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$, learning rate α, period p, local iteration number b
Output: Encrypted global model

@Requester:
Random initialization of the global model ω^0, encrypt ω^0 as $[[\omega^0]]$ with secret key sk, pack $[[\omega^0]]$ and the public key pk to send to *Server*.
for $j = 0$ *to* p **do**
  @Client:
  $\omega^j_{i;0} = [[\omega^0]]$
  foreach $i \in \mathcal{C}$ *Parallel* **do**
    for $k = 1$ *to* b **do**
      $[[\omega_i^j]]_{k+1} \leftarrow [[\omega_i^j]]_k - \alpha \nabla L([[\omega_i^j]]_k; \mathcal{D}_i)$
    end
    Send $[[\omega_i^j]]$ to *Server*.
  end
  @Server:
  Send the ciphertext model $[[\omega^0]]$ and public key pk to client $i, i \in \mathcal{C}$ and wait for the client to return the result after local training.
  $[[\omega_i^{j+1}]] \leftarrow \sum_{i=1}^{|\mathcal{C}|} \frac{1}{|\mathcal{C}|} [[\omega_i^j]]$
  Send the completed training model $[[\omega]]$ to *Requester*.
end
return $[[\omega]]$

6 Preliminary Experiments

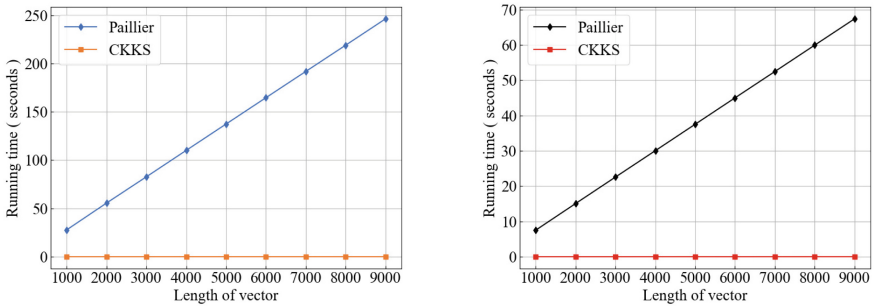
6.1 Experimental Setup

We implement our PriM using the TenSEAL[1] open-source homomorphic encryption library and evaluate the performance with a 128-64-10 fully connected neural network on the MNIST dataset, which contains 60,000 training images and 10,000 test images.

6.2 Runtime Analysis

We compare the runtime of two typical homomorphic schemes, CKKS and Pailier, when dealing with large scale vectors. As can be seen in Fig. 2, CKKS is more suitable for addressing large-scale vectors and models with a great quantity of parameters, because it supports vector encryption without element-wise encryption. Therefore, we adopt CKKS to encrypt the model parameters to improve the computational efficiency.

[1] https://github.com/OpenMined/TenSEAL.

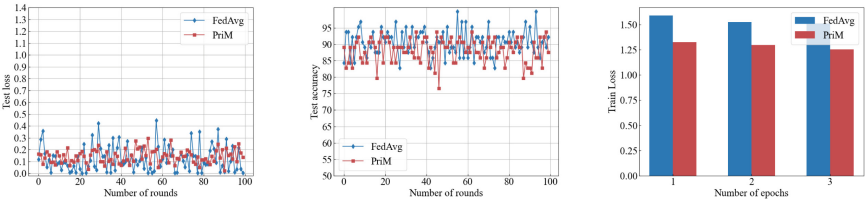


(a) Encryption (b) Decryption

Fig. 2. Computational efficiency of encryption and decryption between CKKS and Paiilier.

6.3 Performance Evaluation

We compare FedAVG [1] as a baseline with PriM for efficiency evaluation. Specifically, because PriM guarantees that the model is not public during the whole Fl process, to check the performance of PriM, we decrypt the loss and prediction values to compare the convergence with the baseline. It's a redundant operation! As shown in Fig. 3, the performance of our PriM is comparable to that of the baseline. Since the CKKS scheme, as an approximate computational homomorphic encryption algorithm, supports approximate addition and multiplication of the model in the ciphertext state, and we also keep four valid digits in our implementation, this leads to a little difference between the computed value after encryption and the original value. It shows that PriM can be more suitable for practical scenarios as an efficient FL model protection scheme.



(a) Convergence analysis (b) Accuracy analysis (c) Training set convergence analysis

Fig. 3. Comparative performance of PriM and FedAVG.

Additionally, PriM deals with massive ciphertext matrix computation in the training phase, while CKKS is a finite-times homomorphic computation scheme; meanwhile, the number of polynomial coefficients generated by encryption determines the depth of homomorphic multiplication and computation delay. Unfortunately, PriM currently overflows the ciphertext size after performing three

rounds of epoch training, and although TenSEAL automatically performs rescaling before the ciphertext multiplication computation, exceeding the homomorphic multiplication depth will result in decryption errors. As shown in Fig. 3, PriM converges with the baseline during training, and the accuracy of the trained model is similar.

7 Conclusion

In this paper, we propose a practical and efficient Federated Learning model preservation framework, PriM. In PriM, the models are confidential to both the server and the client. In contrast to existing research preserving local data, we encrypt the model based on the CKKS fully homomorphic encryption scheme and train and predict it with ciphertext models while follow ing the FL paradigm that private data do not have to be shared. Besides, PriM differs from the traditional PPFL framework without decrypting the model during the whole process, which protects the security of the model. Experiments show that our proposed framework PriM is practical and efficient. In future work, we will employ convolutional neural networks with fewer matrix operations to complete the training of PriM and replace the optimized privacy-preserving measures to ensure the model's confidentiality during the whole FL process.

Acknowledgements. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB3101100; National Natural Science Foundation of China under Grant 62272123, 62272102, 62262058; Project of High-level Innovative Talents of Guizhou Province under Grant [2020]6008; Science and Technology Program of Guiyang under Grant [2021]1–5 and [2022]2–4; Science and Technology Program of Guizhou Province under Grant [2020]5017, [2022]065; Guizhou University Cultivation Project under Grant [2022]22, Guizhou University Talent Introduction Research Fund under Grant GDRJHZ[2015]-53 and Natural Science Foundation of Fujian Province under Grant 2023J02014.

References

1. 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, pp. 1273–1282. PMLR (2017)
2. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 3–18. IEEE (2017)
3. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1322–1333 (2015)
4. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: International Conference on Artificial Intelligence and Statistics, pp. 2938–2948. PMLR (2020)

5. Cao, X., Fang, M., Liu, J., Gong, N.Z.: Fltrust: byzantine-robust federated learning via trust bootstrapping. arXiv preprint arXiv:2012.13995 (2020)
6. 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, pp. 1253–1269 (2020)
7. 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, pp. 1175–1191 (2017)
8. Zhou, Z., Tian, Y., Xiong, J., Ma, J., Peng, C.: Blockchain-enabled secure and trusted federated data sharing in IIoT. IEEE Trans. Ind. Inf. **19**(5), 6669–6681 (2022)
9. Zhou, Z., Tian, Y., Peng, C., Yang, N., Long, S.: VFLF: a verifiable federated learning framework against malicious aggregators in industrial internet of things. Concurr. Comput. Pract. Exp. **35**(20), e7193 (2023)
10. Xu, R., Baracaldo, N., Zhou, Y., Anwar, A., Ludwig, H.: Hybridalpha: an efficient approach for privacy-preserving federated learning. In: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, pp. 13–23 (2019)
11. Wang, Y., Zhang, A., Wu, S., Yu, S.: VOSA: verifiable and oblivious secure aggregation for privacy-preserving federated learning. IEEE Trans. Depend. Secure Comput. **20**(5), 3601–3616 (2022)
12. Stevens, T., Skalka, C., Vincent, C., Ring, J., Clark, S., Near, J.: Efficient differentially private secure aggregation for federated learning via hardness of learning with errors. In: 31st USENIX Security Symposium (USENIX Security 2022), pp. 1379–1395 (2022)
13. Pasquini, D., Francati, D., Ateniese, G.: Eluding secure aggregation in federated learning via model inconsistency. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pp. 2429–2443 (2022)
14. Tian, Y., Wang, S., Xiong, J., Bi, R., Zhou, Z., Bhuiyan, M.Z.A.: Robust and privacy-preserving decentralized deep federated learning training: Focusing on digital healthcare applications. IEEE/ACM Trans. Comput. Biol. Bioinf. **21**, 890–891 (2023)
15. Bi, R., Guo, D., Zhang, Y., Huang, R., Lin, L., Xiong, J.: Outsourced and privacy-preserving collaborative k-prototype clustering for mixed data via additive secret sharing. IEEE Internet Things J. **10**(18), 15810–15821 (2023)
16. Li, T., Tian, Y., Xiong, J., Bhuiyan, M.Z.A.: FVP-EOC: fair, verifiable, and privacy-preserving edge outsourcing computing in 5g-enabled iiot. IEEE Trans. Ind. Inf. **19**(1), 940–950 (2022)
17. Ni, J., Lin, X., Shen, X.S.: Toward edge-assisted internet of things: from security and efficiency perspectives. IEEE Netw. **33**(2), 50–57 (2019)
18. Dapeng, W., Sun, M., Zhang, P., Yanli, T., Yang, Z., Wang, R.: Personalized secure demand-oriented data service toward edge-cloud collaborative iot. IEEE Internet Things J. **10**(1), 378–390 (2022)
19. Luo, C., Ji, J., Wang, Q., Chen, X., Li, P.: Channel state information prediction for 5g wireless communications: a deep learning approach. IEEE Trans. Netw. Sci. Eng. **7**(1), 227–236 (2018)
20. Bi, R., Xiong, J., Tian, Y., Li, Q., Liu, X.: Edge-cooperative privacy-preserving object detection over random point cloud shares for connected autonomous vehicles. IEEE Trans. Intell. Transp. Syst. **23**(12), 24979–24990 (2022)

21. Ma, Z., Ma, J., Miao, Y., Li, Y., Deng, R.H.: ShieldFL: mitigating model poisoning attacks in privacy-preserving federated learning. IEEE Trans. Inf. Forensics Secur. **17**, 1639–1654 (2022)
22. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 409–437. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_15
23. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. **13**(5), 1333–1345 (2017)
24. Xu, R., Joshi, J.B.D., Li, C.: Cryptonn: training neural networks over encrypted data. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 1199–1209. IEEE (2019)
25. Zhao, B., Liu, X., Chen, W.N., Deng, R.H.: CrowdFL: privacy-preserving mobile crowdsensing system via federated learning. IEEE Trans. Mobile Comput. **22**(8), 4607–4619 (2022)
26. Zhao, J., Li, X., Ni, J.: Privacy-preserving model aggregation for asynchronous federated learning. arXiv preprint arXiv:2305.17521 (2023)
27. Miao, Y., Liu, Z., Li, H., Choo, K.K.R., Deng, R.H.: Privacy-preserving Byzantine-robust federated learning via blockchain systems. IEEE Trans. Inf. Forensics Secur. **17**, 2848–2861 (2022)