Accelerating privacy-preserving momentum federated learning for industrial cyber-physical systems

Linlin Zhang^{1,2} · Zehui Zhang³ · Cong Guan⁴

Received: 18 April 2021 / Accepted: 31 August 2021 / Published online: 21 September 2021
© The Author(s) 2021

Abstract

Federated learning (FL) is a distributed learning approach, which allows the distributed computing nodes to collaboratively develop a global model while keeping their data locally. However, the issues of privacy-preserving and performance improvement hinder the applications of the FL in the industrial cyber-physical systems (ICPSs). In this work, we propose a privacy-preserving momentum FL approach, named PMFL, which uses the momentum term to accelerate the model convergence rate during the training process. Furthermore, a fully homomorphic encryption scheme CKKS is adopted to encrypt the gradient parameters of the industrial agents' models for preserving their local privacy information. In particular, the cloud server calculates the global encrypted momentum term by utilizing the encrypted gradients based on the momentum gradient descent optimization algorithm (MGD). The performance of the proposed PMFL is evaluated on two common deep learning datasets, i.e., MNIST and Fashion-MNIST. Theoretical analysis and experiment results confirm that the proposed approach can improve the convergence rate while preserving the privacy information of the industrial agents.

Keywords Federated learning · Deep learning · Privacy preserving · Accelerating convergence

Introduction

Industrial cyber-physical system (CPS) is an emergent technology that focuses on the integration of computational applications with physical devices [1–3]. Industrial CPSs facilitate the remote control of large-scale heterogeneous systems, big data analysis, and condition monitoring, which has a high impact on various industrial fields [2, 4–8]. Industrial CPSs contain many edge devices which collect a huge amount of data, which is very helpful for developing deep learning-based methods to solve difficult industrial tasks, such as fault

diagnosis [9, 10], intelligent control [11], degradation prediction [12], smart city [13], etc. The conventional centralized learning approach (CL) centralizes all the distributed data to a central server for model training. However, if the central server is attacked, the data resources may be revealed, which may result in very bad consequences. In addition, due to business competition and privacy concerns, the data holders (i.e., industrial agents) are unwilling to share their local datasets.

To address this problem, the federated learning (FL) approach is proposed to control multiple training participants to collaboratively train a global model [14, 15]. In the FL system, the training participants only share the gradients of their local models to the cloud server instead of raw data. The central cloud is responsible for updating the global model by aggregating the gradients shared by the training participants and sending the updated global model parameters to all participants. Since the FL can effectively solve data island issues, it has aroused widespread concern in many industrial fields. For instance, Li et al. [16] presented a FL-based CNN-GRU model for intrusion detection that can effectively detect different types of network threats. Kwon et al. [17] proposed a solution for joint cell association and resource allocation in smart ocean scenarios. The authors utilize the FL technologies to meet the requirements of distributed computing and

Linlin Zhang and Zehui Zhang contributed equally to this work.

✉ Zehui Zhang
zhangtianxia918@163.com

¹ Graduate School of Science and Technology, Keio University, Tokyo, Japan

² China Automotive Technology and Research Center Co., Ltd, Tianjin, China

³ College of Software, Nankai University, Tianjin 300071, China

⁴ School of Naval Architecture, Ocean and Energy Power Engineering, Wuhan University of Technology, Wuhan, China

unexpected time-varying states. Liu et al. [18] proposed an FL-based gated recurrent unit neural network for predicting the traffic flow. Brisimi et al. [19] developed a FL-based prediction model to predict the hospitalization of patients suffering from various heart diseases using electronic health records distributed among different data sources. This not only improves the model performance but also ensures the patients' privacy. Zhang et al. [20] proposed a FL-based fault diagnosis method for the rolling bearing of the rotating machinery that combines a dynamic verification scheme and a self-supervised learning scheme.

Various extension methods are proposed to improve the performance of the FL. Yu et al. [21] explained the effectiveness of periodic averaging of a specific model and adopted the parallel mini-batch stochastic gradient descent (SGD) for reducing communication costs. Wang et al. [22] proposed a cooperative SGD framework that combines the periodic-averaging, elastic averaging, and decentralized SGD for simultaneously optimizing the communication cost. Zhao et al. [23] proposed a strategy for globally sharing the small data subsets among all the edge devices to improve the accuracy of FL training on non-IID data. Most of the existing research takes advantage of the first-order gradient descent (GD) for increasing the efficiency of training; however, these techniques do not consider that the previous iterative gradient update potentially accelerates the convergence speed. Liu et al. [24] proposed momentum federated learning (MFL) which uses momentum term to accelerate the convergence during the process of local model training. These works effectively improve the performance of FL; however, they do not consider the privacy concerns appropriately.

Recent studies demonstrate that the FL techniques also suffer from privacy security issues. Wang et al. [25] proposed a framework that combines the generative adversarial networks (GANs) with multitasking discriminators to obtain specific private data from samples without interfering with FL processes. Zhu et al. [26] discussed that the gradient transmission in FL systems may leak the private data without relying on the model generation or prior knowledge of data. Therefore, privacy protection methods for the FL have been proposed. Geyer et al. [27] utilized differential privacy technologies to protect the privacy of distributed participants in the FL system. Triastcyn et al. [28] proposed a Bayesian differential privacy method for FL, which can flexibly adjust the injection noise to provide a stringent privacy guarantee. Aono et al. [29] introduced homomorphic encryption schemes in the FL to protect the data resources of distributed participants. These research works effectively preserve the privacy information in FL under their respective security assumptions. However, they do not consider improving the convergence performance of the FL.

For industrial CPSs, it is necessary that the FL not only performs efficiently in terms of model accuracy, convergence

rate, and communication costs but also preserves the privacy information of the industrial agents. To meet the above requirements, we introduce the momentum term to accelerate the convergence rate of the FL system. Furthermore, the CKKS scheme is used to preserve the privacy information of industrial agents. The contributions of this work are summarized below:

1. We utilize momentum term to accelerate the convergence rate for the privacy-preserving federated learning approach. In particular, the momentum term is calculated by the cloud server to update the global model parameters, which can reduce the cryptography computing and communication costs.
2. To protect the data privacy of the industrial agents, we design a CKKS-based communication scheme, where the industrial agents use the CKKS encryption method to encrypt their gradient parameters.
3. We evaluate the effectiveness of the proposed approach on MNIST and Fashion-MNIST datasets. The experiment results show that the proposed PMFL improves the convergence rate while preserving the data resources of industrial agents.

The rest of this article is organized as follows. In the next section, we introduce the system model and theoretical background. In the subsequent section, we discuss two existing solutions: privacy-preserving federated learning and momentum federated learning followed by which the proposed PMFL approach is presented in detail. Then we analyze the functionality, security and communication costs of the PMFL. The experiment results are presented in the penultimate section. The final section concludes this work and introduces our future research directions.

System model and theoretical background

System model

In this work, the proposed FL system comprises three participants, i.e., trust authority, cloud server, and multiple industrial agents, as shown in Fig. 1.

1. *Trust authority* The trust authority is responsible for ensuring the start-up of the proposed FL system. It produces the public and private keys according to the CKKS-based secure communication protocol. The security communication channels are also established by the trust authority.
2. *Cloud server* The cloud server updates the global model by aggregating all the model parameters uploaded by different industrial agents. It is also responsible for

Fig. 1 The architecture of the FL system

sharing the updated global model with all the industrial agents.

3. **Industrial agents** Each industrial agent trains the model locally using its local dataset and continuously sends the model parameters to the cloud server. After downloading the updated global model, the global weight parameters are loaded as the local models.

Threat model In the proposed model, we assume that the trust authority is completely honest and rational. Hence, it never colludes with the cloud server and the outsiders. In addition, we assume that the cloud server and the industrial agents are honest-but-curious entities who execute the protocols correctly but try to extract additional information by inferring the intermediate data. Based on the assumptions, this work aims to preserve the privacy information of the individual agents during the entire training process (see Table 1).

Full homomorphic encryption

Fully homomorphic encryption (FHE) is a homomorphic technique that can perform multiple mathematical operations on encrypted data [30]. For a typical secure computing scheme, the distributed nodes use FHE to encrypt the data, and the encrypted data are sent to the cloud server. Then the cloud server can perform mathematical operations on the encrypted data without decryption, and send the encrypted results to the distributed nodes. In the scheme, the cloud server is unable to obtain the actual data since the encrypted data cannot be decrypted without the availability of the secret key, and the distributed nodes cannot obtain other nodes' data. CKKS [31] is an FHE-based technique for performing approximate calculation algorithms. The CKKS adds noise after data encryption via ciphertext truncation. Due to this, the resultant scheme has excellent encryption/decryption speed [32]. Therefore,

Table 1 Summary of main notations

Abbreviations	Meaning
CL	Centralized learning
CNN	Convolutional neural networks
CPA	Chosen-plaintext attack
CPSs	Cyber-physical systems
FHE	Fully homomorphic encryption
FL	Federated learning
GANs	Generative adversarial networks
GD	Gradient descent
ICPSs	Industry cyber-physical systems
MFL	Momentum federated learning
MGD	Momentum gradient descent
Non-IID	Non-independent and identically distributed
PPT	Probabilistic polynomial-time
SGD	Stochastic gradient descent
w.r.t	With respect to

we adopt the CKKS-based communication protocol in the proposed method.

Model parameters leak information

In Fig. 2, a sample neural network with three layers is used to illustrate how the model parameters may leak the information of the input data, where x_i denotes the input data, y_i denotes the truth label, w_i denotes the weight parameters of the neural network, and b denotes the bias.

The forward propagation of the network is calculated using the following expression, where $f()$ denotes the sigmoid function.

Fig. 2 A typical neural network model

$$i_{h,1} = \sum_{i=1}^n (x_i \cdot w_i) + b_1, \quad (1)$$

$$o_{h,1} = f(i_{h,1}), \quad (2)$$

The loss value of the model is calculated as

$$L_{\text{total}} = L_{o,1} + L_{o,2} = \frac{1}{2} [(o_{o,1} - y_1)^2 + (o_{o,2} - y_2)^2]. \quad (3)$$

Then, the gradients of the model are calculated as

$$\begin{aligned} \frac{\partial L_{\text{total}}}{\partial w_{1,1}} &= \frac{\partial L_{\text{total}}}{\partial o_{h,1}} \cdot \frac{\partial o_{h,1}}{\partial i_{h,1}} \cdot \frac{\partial i_{h,1}}{\partial w_{1,1}} \\ &= \left(\frac{\partial L_{o,1}}{\partial o_{h,1}} + \frac{\partial L_{o,2}}{\partial o_{h,1}} \right) \cdot o_{h,1} (1 - o_{h,1}) \cdot x_1, \end{aligned} \quad (4)$$

$$\begin{aligned} \frac{\partial L_{\text{total}}}{\partial w_{1,3}} &= \frac{\partial L_{\text{total}}}{\partial o_{h,1}} \cdot \frac{\partial o_{h,1}}{\partial i_{h,1}} \cdot \frac{\partial i_{h,1}}{\partial w_{1,3}} \\ &= \left(\frac{\partial L_{o,1}}{\partial o_{h,1}} + \frac{\partial L_{o,2}}{\partial o_{h,1}} \right) \cdot o_{h,1} (1 - o_{h,1}) \cdot x_2, \end{aligned} \quad (5)$$

$$\frac{\partial L_{\text{total}}}{\partial b_{1,1}} = \frac{\partial L_{\text{total}}}{\partial o_{h,1}} \cdot \frac{\partial o_{h,1}}{\partial i_{h,1}} \cdot \frac{\partial i_{h,1}}{\partial b_{1,1}} = \left(\frac{\partial L_{o,1}}{\partial o_{h,1}} + \frac{\partial L_{o,2}}{\partial o_{h,1}} \right) \cdot o_{h,1} (1 - o_{h,1}) \cdot 1. \quad (6)$$

From Eqs. (4)–(6), we observe that

$$\frac{\partial L_{\text{total}}}{\partial w_{1,1}} / \frac{\partial L_{\text{total}}}{\partial b_{1,1}} = x_1, \quad (7)$$

$$\frac{\partial L_{\text{total}}}{\partial w_{1,3}} / \frac{\partial L_{\text{total}}}{\partial b_{1,1}} = x_2. \quad (8)$$

Therefore, the cloud server can infer the input data x_i of the industrial agent. From Fig. 3, we can notice that a certain percentage of the image may leak its core information.

After the input data x_i is obtained, the model output is calculated through forwarding propagation. The true value of the label y is inferred using Eqs. (9)–(10). Now, the information of the input data is completely leaked.

Fig. 3 Images with different proportions of information leakage

$$\frac{\partial L_{\text{total}}}{\partial w_{2,1}} = \frac{\partial L_{\text{total}}}{\partial o_{o,1}} \cdot \frac{\partial o_{o,1}}{\partial i_{o,1}} \cdot \frac{\partial i_{o,1}}{\partial w_{2,1}} = -(y_1 - o_{o,1}) o_{o,1} (1 - o_{o,1}) o_{h,1}, \quad (9)$$

$$y_1 = o_{o,1} - \frac{\partial L_{\text{total}}}{\partial w_{2,1}} / (o_{o,1} (1 - o_{o,1}) o_{h,1}), \quad (10)$$

$$y_2 = o_{o,2} - \frac{\partial L_{\text{total}}}{\partial w_{2,2}} / (o_{o,2} (1 - o_{o,2}) o_{h,1}). \quad (11)$$

Existing solutions

Privacy-preserving federated learning

By utilizing homomorphic encryption technologies, the privacy-preserving federated learning (PFL) [29] has been proposed to protect the data resources of the training participants. In the PFL system, the training participants calculate the gradients of their local models and encrypt the gradients based on the homomorphic encryption method. Then, the industrial agents upload the encrypted gradients to the cloud server. Using these encrypted gradients, the cloud server updates the global model parameters and sends the encrypted global weight parameters to all the training participants. This is mathematically expressed as

$$\text{Enc}(\mathbf{w}_{\text{global}}) = \text{Enc}(\mathbf{w}_{\text{global}}) - \eta \cdot \frac{\sum_{i=1}^N D_{a,i} \text{Enc}(\mathbf{g}_{a,i})}{\sum_{i=1}^N D_{a,i}}, \quad (12)$$

where $\text{Enc}()$ denotes the encryption function, $\mathbf{w}_{\text{global}}$ denotes the weight parameters of the global model, η denotes the learning rate, $D_{a,i}$ denotes the total number of training samples for training participant i , and $\mathbf{g}_{a,i}$ denotes the gradients of the i th training participant.

Momentum federated learning

Momentum gradient descent algorithm (MGD) is an improved method for the gradient descent algorithm (GD) to speeding up the learning process. In the GD, the update change of

parameters is calculated by $\eta \nabla L(\mathbf{w}(t-1))$ which is only proportional to the gradients of the model. As shown in Fig. 4, the update path of the GD is oscillatory because its update direction is always along gradient descent. In the MGD, the update change of parameters consists of $\eta \nabla L(\mathbf{w}(t-1))$ and $\gamma(\mathbf{w}(t-2) - \mathbf{w}(t-1))$ which is the momentum term to effectively mitigate the oscillation caused by the GD. Momentum term corrects the parameter update direction so that the iterations of the MGD scheme reaching the optimal point are less than that of the gradient descent algorithm. In Fig. 4, the momentum term corrects the parameter update direction so that the iterations of MGD reaching the optimal point are less than that of the GD, which demonstrates that mitigating the oscillation by MGD leads to a faster convergence rate.

Since solutions to convergence acceleration can improve the training efficiency of the FL, Liu et al. [10] proposed Momentum Federated Learning (MFL), where the training participants utilize momentum term to improve the convergence rate. The detailed process is described below.

1. *Initialization* The initial values of momentum parameters $\mathbf{v}_{a,i}(0)$ and model weight parameters $\mathbf{w}_{a,i}(0)$ are initialized.

2. *Industrial agent local training* The MGD algorithm is used by each industrial agent to calculate $\mathbf{v}_{a,i}$ and $\mathbf{w}_{a,i}$ on its respective local dataset, which can be expressed as

$$\mathbf{v}_{a,i}(t) = \gamma \mathbf{v}_{a,i}(t-1) + \nabla L_i(\mathbf{w}_{a,i}(t-1)), \quad (13)$$

$$\mathbf{w}_{a,i}(t) = \mathbf{w}_{a,i}(t-1) - \eta \cdot \mathbf{v}_{a,i}(t), \quad (14)$$

where t denotes the iteration index of local training.

3. *Cloud server parameter aggregation* The cloud server aggregates the parameters upload by the industrial agents to update the global model. The global momentum parameters $\mathbf{v}_{\text{global}}$ and global model parameters weight $\mathbf{w}_{\text{global}}$ are aggregated by taking a weighted average of $\mathbf{v}_{a,i}$ and $\mathbf{w}_{a,i}$. This is expressed in Eqs. (15) and (16). Finally, the global

parameters $\mathbf{v}_{\text{global}}$ and $\mathbf{w}_{\text{global}}$ are transmitted to all the industrial agents for the next training process.

$$\mathbf{v}_{\text{global}}(t) = \frac{1}{|D|} \sum_{i=1}^N |D_{a,i}| \mathbf{v}_{a,i}(t), \quad (15)$$

$$\mathbf{w}_{\text{global}}(t) = \frac{1}{|D|} \sum_{i=1}^N |D_{a,i}| \mathbf{w}_{a,i}(t), \quad (16)$$

where $|D_{a,i}|$ denotes the data size of the industrial agent i .

The proposed framework

In this section, we elaborate on the PMFL framework by introducing the implement procedure for the FL-based application first, and then presenting the workflow of the PMFL, followed by the CKKS-based secure communication scheme.

Implement procedure for the FL-based application

As the FL can solve data barrier issues, it can be utilized to develop deep learning-based applications, which mainly involves five steps as shown in Fig. 5 [15]. First, each industrial agent obtains training samples. Secondly, the agents use the same preprocessing method to process raw data for improving model performance. Then, with the assistance of the cloud server, the industrial agents collaboratively train a global model utilizing their training samples according to the federated learning framework (detailed in Sect. 4.1). Finally, the trained model is evaluated on testing data. In this section, we present the proposed PMFL and its technical details.

Fig. 4 Comparison of the MGD and the GD

Fig. 5 The typical procedures of the FL-based application implementation

The workflow for PMFL

The workflow of the PMFL comprises three phases: system initialization, local model training by industrial agents, and model parameter aggregation executed by the cloud server. The detail of each phase is presented below (see also Fig. 6 and Algorithm 1):

1. *System initialization* The trust authority generates the public key $\mathcal{PK}$ and the private key $\mathcal{SK}$ based on the CKKS encryption, and the cloud server establishes secure communication channels. Each industrial agent initializes the weight parameters of its local model $\mathbf{w}_{a,i}$. Then, each agent encrypts its weights by using the public key. The encrypted parameters are transmitted to the cloud server. Then the cloud server computes the global model by aggregating all the uploaded parameters. Finally, the server sends the initialized global model $\text{Enc}(\mathbf{w}_{\text{global}})$ to all the industrial agents.

2. *Local model training by industrial agents* After receiving the encrypted global model parameters, i.e.,

$\text{Enc}(\mathbf{w}_{\text{global}})$, from the cloud server, each industrial agent uses the private key $\mathcal{SK}$ to decrypt the weight parameters $\text{Enc}(\mathbf{w}_{\text{global}})$ and obtains $\mathbf{w}_{\text{global}}$. After model decryption, the agent loads $\mathbf{w}_{\text{global}}$ in its local model, and calculates the gradients of the local model $\mathbf{g}_{a,i}$ using the private data resource. Lastly, the agent uses the public key to encrypt the gradients and sends $\text{Enc}(\mathbf{g}_{a,i})$ to the cloud server.

3. *Model parameter aggregation by the cloud server* After receiving the encrypted gradients $\text{Enc}(\mathbf{g}_{a,i})$ from each industrial agent, the cloud server calculates the encrypted global momentum term $\text{Enc}(\mathbf{v}_{\text{global}}(t))$ using Eq. (17), where $n_{a,i}$ denotes the total number of training samples for industrial agent i . Then, the encrypted global parameters $\text{Enc}(\mathbf{w}_{\text{global}}(t))$ are updated using Eq. (18), and sent $\text{Enc}(\mathbf{w}_{\text{global}}(t))$ the parameters to the industrial agents.

$$\text{Enc}(\mathbf{v}_{\text{global}}(t)) = \gamma \text{Enc}(\mathbf{v}_{\text{global}}(t-1)) + \frac{\sum_{i=1}^N n_{a,i} \text{Enc}(\mathbf{g}_{a,i})}{\sum_{i=1}^N n_{a,i}}, \quad (17)$$

$$\text{Enc}(\mathbf{w}_{\text{global}}(t)) = \gamma \text{Enc}(\mathbf{w}_{\text{global}}(t-1)) + \eta \text{Enc}(\mathbf{v}_{\text{global}}(t)). \quad (18)$$

Fig. 6 The workflow of the proposed PMFL

Algorithm 1: Privacy Momentum Federated Learning

Input: Epoch, momentum γ , mini-batch n_b , industrial agent ia_k
Output: FL-based model

1 Initialization:
 2 a). The trust authority generates $\mathcal{PK}$ and $\mathcal{SK}$;
 3 b). Establish secure communication channels;
 4 c). Initialize the global model $\mathbf{w}_{\text{global}}$;
4 Procedure:
 5 **for** $e \leq \text{Epoch}$ **do**
 6 $t = 0$
 7 **for** $t \leq T$ **do**
 8 **(I). For industrial agents:** // Parallel computing
 9 **for** $i \leq N$ **do**
 10 ia_k calculates the gradients of the local model $\mathbf{g}_{a,i}(t)$;
 11 ia_k encrypts the gradients of the local model $\mathbf{g}_{a,i}(t)$ based on CKKS encryption scheme (Section 4.2);
 12 ia_k sends the encrypted parameters $\text{Enc}(\mathbf{g}_{a,i}(t))$ to the cloud server ;
 13 **end**
 14 **(II). For cloud server:**
 15 Receive $\text{Enc}(\mathbf{g}_{a,i}(t))$ from all industrial agents;
 16 Update the global model according to Eqs. (17) and (18);
 17 Send the encrypted global model $\text{Enc}(\mathbf{w}_{\text{global}}(t))$ to all industrial agents;
 18 **(III). For industrial agents:** // Parallel computing
 19 **for** $k \leq N$ **do**
 20 ia_k decrypt the encrypted parameters $\text{Enc}(\mathbf{w}_{\text{global}}(t))$ based on CKKS encryption scheme (Section 4.2);
 21 ia_k loads the global model parameters as the local model;
 22 **end**
 23 **end**
 24 **end**
 25 **return** FL-based model $\mathbf{w}_{\text{global}}$

CKKS-based secure communication protocol

Various research works [33–35] indicated that the encryption/decryption operations of homomorphic encryption are computationally expensive. However, the CKKS as an emerging encryption scheme has excellent encryption speed compared to the Paillier and RSA encryption schemes [36, 37]. To improve the training efficiency of the FL system, the proposed approach uses the CKKS scheme to preserve the data resources of the industrial agents.

In the PMFL, the CKKS encryption scheme is used to encrypt the model parameters. $\mathcal{R}$ represents a polynomial ring $\mathbb{Z}[X]/(X^n + 1)$. We use $\mathcal{R}$ to denote $\mathbb{Z}[X]/(X^n + 1)$ with integer coefficients modulo $\mathcal{R}q$. Similarly, we use $[x]_q$ to denote $x \bmod q$. Please note that $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes the inner product of vectors $\mathbf{a}$ and $\mathbf{b}$. The CKKS-based secure communication protocol includes the following four functions (detailed in [31]).

1. *KeyGenerate*(1^λ) Given a security parameter λ , we set a ring degree n , a ciphertext modulus q , and a special modulus p which is coprime of q . The trust authority generates the secret key $\mathcal{SK}$ and the public key $\mathcal{PK}$ as per the standard defined by CKKS cryptosystem.

2. *ParaEncrypt*($m, \mathcal{PK}$): We sample $v \leftarrow \chi_{\text{enc}}$ and $e_0, e_1 \leftarrow \chi_{\text{err}}$, and encrypt the model parameter using the public key $\mathcal{PK}$ by

$$\text{Enc}(m) \leftarrow [v \cdot \mathcal{PK} + (m + e_0, e_1)]_{q_L} \in \mathcal{R}_{q_L}^2, \quad (19)$$

where m denotes the plaintext.

3. *ParaAggregate*($\text{Enc}(m_1), \dots, \text{Enc}(m_N)$): The cloud server aggregates the encrypted parameters $\text{Enc}(m_k) \in \mathcal{R}_{q_L}^2$ to update the momentum or global model parameters by

$$\text{Enc}(m_{\text{agg}}) \leftarrow \text{sum}[\text{Enc}(m_1), \dots, \text{Enc}(m_N)]_{q_L} \in \mathcal{R}_{q_L}^2. \quad (20)$$

4. *ParaDecrypt*($\text{Enc}(m_{\text{agg}}), \mathcal{SK}$): The industrial agents use the secret key $\mathcal{SK}$ to decrypt the encrypted parameters of the

Table 2 Functionality analysis of the recent FLs and PMFL

Functionality\Approach	DPFL	PFL	MFL	PMFL
Gradients confidentiality	✓	✓	✓	✓
Privacy Protection	✓	✓	×	✓
Noninteractive protocol	×	✓	×	✓
Momentum term utilization	×	×	✓	✓

global model sent by the cloud server using the following technique.

$$m \leftarrow [\langle \text{Enc}(m_{agg}), \mathcal{SK} \rangle]_{q_L}. \quad (21)$$

Algorithm analysis

Functionality analysis

In Table 2, we present the functional comparison of various FL techniques, such as DPFL [28], PFL [29], and MFL [24]. The DPFL and the PFL preserve the privacy of the industrial agents. However, both solutions do not use the momentum parameters to accelerate the rate of convergence. Contrary, the MFL utilizes the momentum to accelerate the training process; however, it does not preserve the privacy of the industrial agents. The proposed PMFL incorporates the momentum parameters to improve the convergence rate while preserving the privacy of the industrial agents using the CKKS encryption scheme.

Security analysis

CPA-secure: For each probabilistic polynomial-time (PPT) and an adversary $\mathcal{A}$, there is a negligible function negl , such that

$$\Pr[\text{privK}_{A,\Pi}^{\text{cpa}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n), \quad (22)$$

$\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ is a CPA security encryption method in which the probability is provided by the randomness of the adversary $\mathcal{A}$ and the randomness of the experiment (the generated $\mathcal{PK}$, random bit, and any randomness used in the encryption process). Based on this definition, we obtain the following:

All the encryption techniques that meet the CPA security, can guarantee security even in the presence of eavesdroppers.

All the deterministic encryption schemes do not satisfy the CPA security, and the encryption scheme that fulfills the CPA security must be probabilistic encryption.

We assume that the commutation channel between each industrial agent and the cloud server is sufficiently secure. This allows the server to verify the integrity of the uploaded data and prevent the potential attackers from performing any malicious activity, such as injecting their own data. The intermediate data obtained by the industrial agents and the cloud server during the training process are presented in Table 3. It is noteworthy that during the training process, each industrial agent can only obtain the encrypted weight parameters of the global model $\text{Enc}(\mathbf{w}_{\text{global}})$ from the cloud server, the cloud server only receives the encrypted gradient parameters of the models $\text{Enc}(\mathbf{g}_{\text{global}})$ from each industrial agent, and sends back the encrypted global model parameters $\text{Enc}(\mathbf{w}_{\text{global}})$.

Theorem 1 *In the PMFL, if CKKS is CPA-secure, and there is no collusion between industry agents and the cloud server/external attacker, the data privacy of the industrial agents can be preserved.*

Proof Assume that there is an adversary $\mathcal{A}$ who eavesdrops on the encrypted weight parameters of all the models. Since the adversary does not know the security parameter λ of the CKKS scheme, $\mathcal{A}$ cannot generate the secret key $\mathcal{SK}$. Based on the above security assumptions, all industrial agents do not collude with the cloud server or any external member, and $\mathcal{SK}$ will not be leaked. Therefore, $\mathcal{A}$ is unable to obtain the key $\mathcal{SK}$ and decrypt the encrypted parameters to obtain the true values. At the same time, the weight parameters of the model are stored on the cloud server in the ciphertext form. As long as the cloud server does not conspire with any industrial agents, each industrial agent cannot obtain the model parameters uploaded by other industrial agents for inferring their data resources. In addition, the agents transmit information to the cloud server through different secure communication channels, thereby preventing the transmitted information from being stolen. Therefore, the proposed

Table 3 Data information obtained by the industrial agents and cloud server

Name	Industrial agents	Cloud server
Intermediate data	$\text{Enc}(\mathbf{w}_{\text{global}})$	$\text{Enc}(\mathbf{g}_{\text{global}})$
	$L_{a,i}$	$\text{Enc}(\mathbf{g}_{a,1})$
	$\mathbf{g}_{a,i}$	...
	$\mathbf{w}_{a,i}$	$\text{Enc}(\mathbf{g}_{a,n})$

Table 4 The communication cost of PMFL compared with MFL

Framework	Each industrial agent	Cloud server
PMFL	$O(\text{Enc}(\mathbf{g}_{a,i})) + O(\text{Enc}(\mathbf{w}_{\text{global}}))$	$O((\text{Enc}(\mathbf{g}_{a,i}) \cdot N) + O(\text{Enc}(\mathbf{w}_{\text{global}})) \cdot N)$
MFL	$O(\mathbf{v}_{a,i} + \mathbf{w}_{a,i}) + O(\mathbf{v}_{\text{global}} + \mathbf{w}_{\text{global}})$	$O((\mathbf{v}_{a,i} + \mathbf{w}_{a,i}) \cdot N) + O((\mathbf{v}_{\text{global}} + \mathbf{w}_{\text{global}}) \cdot N)$

Table 5 The structure parameters of the CNN model

Model	CNN
Structure	Conv2d (1,16, kernel_size=5, padding=1) MaxPool (2×2) Relu () Conv2d (16,32, kernel_size=5, padding=1) MaxPool (2×2) Relu () FC (800,64) with Relu activation function Full connected layer (64,10) with softmax layer

PMFL can effectively protect the private information of the industrial agents.

Communication cost analysis

We assume that the FL system includes N industrial agents. We set the length of $\mathbf{w}_{\text{global}}$ and $\mathbf{g}_{a,i}$ without considering the increase in the transmission cost of the ciphertext. The communication costs of the PMFL is further analyzed and compared with the MFL.

Communication cost of the proposed PMFL For each round of model aggregation, the industrial agents send $\text{Enc}(\mathbf{g}_{a,i})$ to the cloud server and receive $\text{Enc}(\mathbf{w}_{\text{global}})$ from the cloud server. The communication costs of the cloud server and each industrial agent are $O((|\text{Enc}(\mathbf{g}_{a,i})| \cdot N) + O(|\text{Enc}(\mathbf{w}_{\text{global}})|) \cdot N)$ and $O(|\text{Enc}(\mathbf{g}_{a,i})|) + O(|\text{Enc}(\mathbf{w}_{\text{global}})|)$, respectively.

Communication cost for the MFL For each model aggregation, each industrial agent sends $\mathbf{v}_{a,i}$ and $\mathbf{w}_{a,i}$ to the cloud

server and receives global $\mathbf{v}$ and $\mathbf{w}$ from the cloud server. The communication costs of the cloud server and each industrial agent are $O((|\mathbf{v}_{a,i}| + |\mathbf{w}_{a,i}|) \cdot N) + O((|\mathbf{v}_{\text{global}}| + |\mathbf{w}_{\text{global}}|) \cdot N)$ and $O(|\mathbf{v}_{a,i}| + |\mathbf{w}_{a,i}|) + O(|\mathbf{v}_{\text{global}}| + |\mathbf{w}_{\text{global}}|)$, respectively.

In Table 4, we present a comparison of the communication cost of the proposed PMFL and the MFL. The above analyses show that the communication cost of the PMFL is lower than the MFL.

Performance evaluation

In this section, we present the simulation results to evaluate the performance of the proposed PMFL method. We first present the simulation setup, which includes the environmental setup, data resource description, and performance metrics. Then, we evaluate the proposed PMFL.

Experiment settings

The hardware operating environment is Intel CPU i7-6550, RAM 16 GB, and NVIDIA 1080Ti. We build the FL system using Python, NumPy, Pytorch, MATLAB, and CUDA. The MNIST dataset and Fashion-MNIST (F-MNIST) are used to evaluate the performance of the proposed FL system. The MNIST dataset contains 60,000 training samples and 10,000 test samples. It has 10 categories, and each sample is a 28×28 grayscale image. The F-MNIST data are similar to the MNIST dataset.

In this work, the FL system includes four industrial agents. To simulate real industrial scenarios, the datasets

Fig. 7 The experiment curves of different learning approaches**Table 6** The classification results of the different learning approaches

Dataset	F-MNIST			MNIST		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
CL	87.00%	87.36%	87.00%	98.49%	98.49%	98.49%
PFL	83.03%	83.64%	83.03%	97.99%	98.00%	97.97%
MFL	86.24%	86.67%	86.24%	98.16%	98.16%	98.15%
PMFL	86.53%	86.93%	86.53%	98.13%	98.16%	98.13%

are split into non-independent and identically distributed (Non-IID) datasets, which indicates that all data held by an industrial agent have the same label (if there are more labels than the number of the industrial agents, each industrial agent have data with more than one label but not the total number of labels). That is in accordance with the real industrial scenarios since the training data are generally difficult to be achieved.

Convolutional neural network (CNN) has been widely used in computer vision and image recognition fields. Therefore, referring to Pytorch tutorials, a CNN model is selected as the local model of the industrial agents for evaluating the performance of the proposed PMFL approach. The construction parameters of the CNN are listed in Table 5. The mini-batch size of the industrial agents is set to 64.

Case 1 (comparisons of different learning approaches)

In this subsection, we evaluate the performance of CL, FL, MFL and PMFL. The experimental curves and classification results of different learning approaches are presented in Fig. 7 and Table 6, respectively. In general, there is a performance gap between the FL and the CL. It can be seen from Fig. 7 that the accuracy and the loss value curves of the PMFL are closer to the CL compared to the FL. As listed in Table 6, the PMFL achieves the classification accuracies of 86.53% and 98.13% on F-MNIST dataset and MNIST dataset, respectively. The classification results of the PMFL are very close to the CL and superior to the FL. Compared with the MFL, the PMFL not only has almost the same

performance but also saves half of the communication cost between the cloud server and the industrial agents.

Case 2 (varying experiment setting)

In this subsection, we verify the convergence performance of the PMFL and explore the effects of the momentum rate γ and the number of the industrial agents N on the PMFL convergence by simulation evaluation.

1 Varying momentum rates In this subsection, the experiment curves of the proposed PMFL with different momentum rates are presented in Fig. 8. When the momentum rate γ is set to 0, the PMFL is the same as the traditional PFL. From the figure, it can be noticed that the accuracy curves with different momentum rates have a consistent overall trend. The convergence speed of PMFL is improved with the increasing momentum rate. The classification results of the proposed PMFL with different momentum rates are listed in Table 7. When $\gamma=0.6$, the PMFL has the best classification performance on F-MNIST and MNIST datasets. The classification results reveal that the classification performance of the PEMFL is affected by the momentum rate. Therefore, the momentum rate value needs to be appropriately selected instead of just a large value.

2. Varying numbers of industrial agents For this case, we set the momentum rate γ to 0.5 and varies the number of industrial agents N . The experiment curves of the PMFL with different numbers of the industrial agents are shown in Fig. 9. The classification results of the proposed PMFL with different N are listed in Table 8. The trend of the curves shows the convergence performance of the PMFL. It can

Fig. 8 The experiment curves of the PMFL with different momentum rates

Table 7 The classification results of PMFL with different momentum rates

Dataset	F-MNIST			MNIST		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
PMFL($\gamma=0$)	83.03%	83.64%	83.03%	97.99%	98.00%	97.97%
PMFL($\gamma=0.3$)	86.44%	86.29%	86.44%	98.26%	98.27%	98.25%
PMFL ($\gamma=0.6$)	86.64%	86.54%	86.64%	98.43%	98.42%	98.42%
PMFL ($\gamma=0.9$)	86.63%	86.54%	86.63%	97.60%	97.64%	97.54%

Fig. 9 The classification results of the CNN-based PMFL with different numbers of Industrial Agents

Table 8 Results of PMFL with different numbers of the industrial agents N

Dataset	F-MNIST			MNIST		
	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)
PMFL($N=4$)	86.53	86.93	86.53	98.13	98.16	98.13
PMFL($N=8$)	84.51	84.43	84.51	97.96	97.94	97.96
PMFL($N=16$)	82.79	82.98	82.98	97.00	96.99	96.97
PMFL($N=32$)	80.09	80.05	80.09	95.91	95.90	95.86

Bold values indicate the best result

Fig. 10 The learning rate tuning results of the PMFL on F-MNIST and MNIST datasets

Fig. 11 The momentum rate tuning of the PMFL on F-MNIST and MNIST datasets

be noticed that when N increases from 4 to 32, the PMFL achieves the classification accuracies of 80.09–86.53% and 95.91–98.13% on F-MNIST dataset and MNIST dataset, respectively. Further analysis of Table 8 shows that the

convergence speed of the PMFL slows as the number N increases, because the model aggregation process utilizes more data. Consequently, the PMFL has a bright prospect in the CPSs with multiple industrial agents.

Case 3 (hyperparameter tuning)

Liking most deep learning tasks, the hyperparameters of the CNN trained by the FL need to be tuned for improving model accuracy. In this subsection, the first hyperparameter to tune is the learning rate η . We set momentum rate γ to 0.5 and mini-batch to 64. The learning rate η is set to be from 0.01 to 0.1. The learning rate tuning results of the PMFL are shown in Fig. 10.

The second hyperparameter to tune is the momentum rate γ . From the above results, we set the learning rate $\eta=0.07$ for the F-MNIST task and $\eta=0.08$ for the MNIST task. The momentum rate γ is set to be from 0.1 to 0.9. The learning rate tuning results of the PMFL are shown in Fig. 11. From the above results, we set the momentum rate $\gamma=0.5$ for the F-MNIST task and $\gamma=0.7$ for the MNIST task.

Based on the above results, we set learning rate η to 0.08, momentum rate γ to 0.7 and mini-batch size to 64 for MNSIT task, and learning rate η to 0.07, momentum rate γ to 0.5 and mini-batch size to 64 for MNSIT task.

Conclusions

In this work, we present a privacy-preserving momentum federated learning (PMFL) for industrial cyber-physical systems (ICPSs). In the proposed FL approach, a CKKS-based secure communication protocol is designed to guarantee the privacy of the industrial agents by encrypting the weight parameters of the local models. The momentum term is utilized in the PMFL to accelerate the convergence rate. In particular, the momentum term is calculated in the cloud server, which reduces the communication costs compared to the MFL. Theoretical analysis and experiment results demonstrate that the proposed approach can effectively preserve the local privacy of the industrial agents and has high accuracy as well.

In future work, we aim to further investigate the process of model aggregation for alleviating the adverse effects of the industrial agents with low-quality training data, and use intelligent algorithms to tune hyper-parameters of the FL to achieve better model performance.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (51909200), International S&T Cooperation Program of China (2019YFE0104600), Tianjin Research Innovation Project for Postgraduate Students (2019YJSB06).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are

included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Huang S, Lin C, Zhou K, Yao Y, Lu H, Zhu F (2020) Identifying physical-layer attacks for IoT security: An automatic modulation classification approach using multi-module fusion neural network. *Phys Commun* 43:101180
- Ding D, Han Q, Wang Z, Ge X (2019) A survey on model-based distributed control and filtering for industrial cyber-physical systems. *IEEE Trans Ind Inf* 15:2483–2499
- Yang F, Gu S (2021) Industry 4.0, a revolution that requires technology and national strategies. *Complex Intel Syst* 7(3):1–15
- Chen H, Zhang Z, Guan C, Gao H (2020) Optimization of sizing and frequency control in battery/supercapacitor hybrid energy storage system for fuel cell ship. *Energy* 197:117285
- Zhang Z, Guan C, Liu Z (2020) Real-time optimization energy management strategy for fuel cell hybrid ships considering power sources degradation. *IEEE Access* 8:87046–87059
- Khalid A, Kirisci P, Khan ZH, Ghrairi Z, Thoben K, Pannek J (2018) Security framework for industrial collaborative robotic cyber-physical systems. *Comput Ind* 97:132–145
- Khalifa AH, Shehata MK, Gasser SM, El-Mahallawy MS (2020) Enhanced cooperative behavior and fair spectrum allocation for intelligent IoT devices in cognitive radio networks. *Phys Commun* 43:101190
- Huo W, Li W, Zhang Z, Sun C, Zhou F, Gong G (2021) Performance prediction of proton-exchange membrane fuel cell based on convolutional neural network and random forest feature selection. *Energy Convers Manag* 243:114367
- Gong W, Chen H, Zhang Z, Zhang M, Wang R, Guan C, Wang Q (2019) A novel deep learning method for intelligent fault diagnosis of rotating machinery based on improved CNN-SVM and multichannel data fusion. *Sensors* 19:1693
- Zhang J, Huang K (2020) Fault diagnosis of coal-mine-gas charging sensor networks using iterative learning-control algorithm. *Phys Commun* 43:101175
- Lan Y, Li F, Li Z, Yue B, Zhang Y (2020) Intelligent IoT-based large-scale inverse planning system considering postmodulation factors. *Complex Intell Syst*, pp 1–15
- Zuo B, Cheng J, Zhang Z (2021) Degradation prediction model for proton exchange membrane fuel cells based on long short-term memory neural network and Savitzky-Golay filter. *Int J Hydrog Energy* 46(29):15928–15927
- Gomathi P, Baskar S, Shakeel PM (2020) Concurrent service access and management framework for user-centric future internet of things in smart cities. *Complex Intell Syst* 7:1723–1732
- Zhu H, Zhang H, Jin Y (2020) From federated learning to federated neural architecture search: a survey. *Complex Intell Syst* 7:639–657
- Zhang Z, Xu X, Gong W, Chen Y, Gao H (2021) Efficient federated convolutional neural network with information fusion for rolling bearing fault diagnosis. *Control Eng Pract* 116:104913
- Li B, Wu Y, Song J, Lu R, Li T, Zhao L (2020) DeepFed: Federated deep learning for intrusion detection in industrial cyber-physical systems. *IEEE Trans Ind Inf* 17:5615–5624
- Kwon D, Jeon J, Park S, Kim J, Cho S (2020) Multiagent DDPG-Based Deep Learning for Smart Ocean Federated Learning IoT Networks. *IEEE Internet Things J* 7:9895–9903

18. Liu Y, James JQ, Kang J, Niyato D, Zhang S (2020) Privacy-preserving traffic flow prediction: a federated learning approach. *IEEE Internet Things J* 7:7751–7763
19. Brisimi TS, Chen R, Mela T, Olshevsky A, Paschalidis IC, Shi W (2018) Federated learning of predictive models from federated electronic health records. *Int J Med Inf* 112:59–67
20. Zhang W, Li X, Ma H, Luo Z, Li X (2021) Federated learning for machinery fault diagnosis with dynamic validation and self-supervision. *Knowl-Based Syst* 213:106679
21. Yu H, Yang S, Zhu S (2019) Parallel restarted SGD with faster convergence and less communication: demystifying why model averaging works for deep learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp 5693–5700
22. Wang J, Joshi G. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms[C]/*ICML Workshop on Coding Theory for Machine Learning*. 2019.
23. Zhao Y, Li M, Lai L, Suda N, Civin D, Chandra V (2018) Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*
24. Liu W, Chen L, Chen Y, Zhang W (2020) Accelerating Federated Learning via Momentum Gradient Descent. *IEEE Trans Parallel Distrib Syst* 31:1754–1766
25. Wang Z, Song M, Zhang Z, Song Y, Wang W, Qi H (2019) Beyond inferring class representatives: user-level privacy leakage from federated learning. In: *IEEE INFOCOM 2019-IEEE Conference on computer communications*, IEEE, 2019, pp 2512–2520
26. Zhu L, Han S (2020) Deep Leakage from Gradients. In: Yang Q, Fan L, Yu H (eds) *Federated Learning*. Lecture Notes in Computer Science, Vol. 12500. Springer, Cham. https://doi.org/10.1007/978-3-030-63076-8_2
27. RGeYer RC, Klein T, Nabi M (2017). Differentially private federated learning: a client level perspective. *arXiv preprint arXiv:1712.07557*
28. Triastcyn A, Faltings B (2019) Federated learning with Bayesian differential privacy. In: *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, 2019, pp 2587–2596
29. Aono Y, Hayashi T, Phong LT, Wang L (2016) Privacy-preserving logistic regression with distributed data sources via homomorphic encryption. *IEICE Trans Inf Syst*, 99(8):2079–2089
30. Sumathi M, Sangeetha S (2020) A group-key-based sensitive attribute protection in cloud storage using modified random Fibonacci cryptography. *Complex Intell Syst*, pp 1–15
31. Cheon JH, Kim A, Kim M, Song Y (2017) Homomorphic encryption for arithmetic of approximate numbers. In: Takagi T, Peyrin T (eds) *Advances in Cryptology-ASIACRYPT 2017*. ASIACRYPT 2017. Lecture Notes in Computer Science, Vol. 10624. Springer, Cham. https://doi.org/10.1007/978-3-319-70694-8_15
32. Ou W, Zeng J, Guo Z, Yan W, Liu S, Fuentes S (2020) Ahomomorphic-encryption-based vertical federated learning scheme for rick management. *Comput Sci. Inf Syst* 17(3):819–834
33. Hao M, Li H, Xu G, Liu S, Yang H (2019) Towards efficient and privacy-preserving federated deep learning. In: *ICC 2019–2019 IEEE International Conference on Communications (ICC)*, IEEE, 2019, pp 1–6
34. Li L, Fan Y, Tse M, Lin K (2020) A review of applications in federated learning. *Comput Ind Eng* 149:106854
35. Zhou C, Fu A, Yu S, Yang W, Wang H, Zhang Y (2020) Privacy-preserving federated learning in fog computing. *IEEE Internet Things J* 7:10782–10793
36. Chen H, Dai W, Kim M, et al. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference[C]. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019: 395–412.
37. Liu X, Li H, Xu G, Liu S, Liu Z, Lu R (2020) PADL: privacy-aware and asynchronous deep learning for IoT applications. *IEEE Internet Things J* 7:6955–6969

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.