Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-02664-x

ORIGINAL RESEARCH

Verifiable and privacy preserving federated learning without fully trusted centers

Gang Han[1] · Tiantian Zhang[1] · Yinghui Zhang[1] · Guowen Xu[2] · Jianfei Sun[2] · Jin Cao[3]

Received: 25 June 2020 / Accepted: 3 November 2020

Abstract

With the rise of neural network, deep learning technology is more and more widely used in various fields. Federated learning is one of the training types in deep learning. In federated learning, each user and cloud server (CS) cooperatively train a unified neural network model. However, in this process, the neural network system may face some more challenging problems exemplified by the threat of user privacy disclosure, the error of server's returned results, and the difficulty of implementing the trusted center in practice. In order to solve the above problems simultaneously, we propose a verifiable federated training scheme that supports privacy protection over deep neural networks. In our scheme, the key exchange technology is used to remove the trusted center, the double masking protocol is used to ensure that the privacy of users is not disclosed, and the tag aggregation method is used to ensure the correctness of the results returned by the server. Formal security analysis and comprehensive performance evaluation indicate that the proposed scheme is secure and efficient.

Keywords Federated learning · Privacy protection · Deep neural network · Certificateless signature

1 Introduction

Deep neural network is a kind of network model in the field of machine learning. The artificial intelligence (AI) technology it represents is considered as one of the cornerstones of technological change (Sze et al. 2017). In recent years, deep neural network has made outstanding progress in the field of speech and image. Deep learning can be understood as using deep neural networks for machine learning. Deep learning is of great research value in many areas (Mohammadi et al. 2018). For example, in medical health, deep learning can be used to classify and analyze medical pictures (Liu et al. 2016); in intelligent traffic systems, deep learning can be used to ameliorate the automatic driving vehicle system (Ma et al. 2015).

In supervised machine learning, some data is used for verification (Gu et al. 2020). Generally speaking, as a type of machine learning, deep learning needs to collect massive amounts of data from users. These data are accumulated unsupervised in the accumulation process (García-Gil et al. 2019) and used for analysis and processing. However, the data provided by users inevitably contains some sensitive information of users. As mentioned earlier, in the medical environment, deep learning is used in assisted diagnosis and treatment, but in this process, patients' privacy will

✉ Tiantian Zhang
ttzhng@163.com

Gang Han
hangang668866@163.com

Yinghui Zhang
yhzhaang@163.com

Guowen Xu
guowen.xu@foxmail.com

Jianfei Sun
sjf215.uestc@gmail.com

Jin Cao
caoj897@gmail.com

[1] School of Cyberspace Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

[2] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

[3] State Key Laboratory of Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi'an 710071, China

Published online: 03 January 2021

inevitably be affected (Guowen et al. 2019). For patients, naturally, they are reluctant to disclose their condition. Hence, data privacy issue is a hot research topic in deep learning. Federated training is a kind of training in which all users and the CS work collaboratively by sharing local gradients and global parameters, and can train the network's ability without collecting user's original data.

In order to strengthen data privacy and security, in 2016, Google first proposed a possible solution, the Federal Learning Framework (Yang et al. 2019). Generally speaking, in deep learning, the technical ability of each AI enterprise is single combat. With the emergence of federated learning, all artificial intelligence enterprises are more closely and safely linked together. Each member of the Federation can use the fastest speed to improve their own capabilities and learn from the strengths of others, and finally achieve common growth (Dong 2019).

Due to hardware defects, software vulnerabilities and improper human operations, data is threatened with damage in the age of information technology. Data integrity has also become an issue that must be considered in the training process of federated learning. For the sake of ensuring data integrity, some data integrity detection schemes are presented (Zhang et al. 2017; Yu et al. 2016; Shen et al. 2017; Jiang et al. 2015; Wang et al. 2015; Shen et al. 2019; Zhao et al. 2019; Shao et al. 2018; Wang et al. 2011; Yan et al. 2016, 2019).

Ensuring the privacy of user data and proving the integrity of the results returned from the CS are the two major challenges facing federated training over deep neural networks. Hence, it makes sense to consider a verifiable federated training scheme that supports data privacy protection in deep learning.

1.1 Related work

As deep learning is applied to safety-critical or sensitive fields, the privacy and integrity of data are becoming increasingly demanding.

Some research schemes have been proposed for data privacy protection (Hitaj et al. 2017; Phong et al. 2018; Moriai 2019; Zhao et al. 2020; Li et al. 2018). Hitaj et al. conducted research on the information leakage problem of collaborative deep learning (Hitaj et al. 2017). The research reveals that adversaries may obtain sensitive information indirectly based on sharing gradient. Phong et al. (2018) realized that participating users perform deep learning based on neural networks on a combination of all data sets, without divulging the local data of participants to the CS, aiming to design a privacy-protected deep learning system. Moriai (2019) carried out research on privacy protection data analysis, such as privacy protection logic regression and privacy protection deep learning, linking deep learning and cryptography to develop a privacy-preserving financial data analysis system with high security and accuracy fraud detection. Zhao et al. (2020) designed a collaborative deep learning framework to effectively prevent the leakage of private training data. Besides, Li et al. (2018) use double decryption technology to protect privacy.

Furthermore, Shokri and Shmatikov (2015) designed an accurate neural network system model in which multiple parties can learn a given goal together without sharing the input data set. The system allows users to train independently on their own datasets, and selectively share the subset of key parameters of the model during the training. Chen et al. (2020) pointed out that there may be malicious users injecting wrong training results through causal attacks in order to destroy the learning model, so a training integrity protocol that can detect causal attacks is proposed. To address the high communication cost caused by frequent interaction between data owners during the training process, Li et al. (2020) designed a non-interactive privacy protection multi-party machine learning framework.

At present, many scholars have studied the integrity of data. Nan et al. (2019) proposed collusion-resistant integrity verification by taking advantage of aggregating BLS signature technology. Fan et al. (2019) put forward a secure data integrity verification scheme on account of the security identity of the aggregated signature. Tian et al. (2019) proposed a batch integrity audit strategy, which supports shared-data dynamics. This strategy further extends the dynamic hash table. Besides, Zhang et al. presented two block-based fair payment protocols (Zhang et al. 2018a, b), which can perform two-side verification of trusted keyword search (Zhang et al. 2018), secure outsourcing computation and dynamic data integrity.

In addition, motivated by the above requirements, further related research is as follows. Bonawitz et al. (2017) designed a high-dimensional data security aggregation protocol for federated learning. The protocol can be used to aggregate user-provided deep neural network model updates, and it has been shown that any user subset can remain secure at any time when it exits. Guowen et al. (2020) indicates that the scheme does not support the verification of the correctness of the results, which go back from the CS. Hence, they proposed the unprecedented privacy protection and verifiable federal learning skeleton frame, VerifyNet, which uses a double masking protocol to guarantee the confidentiality of data. The framework also supports users to withdraw during the training process. In addition, the program uses data to conduct a large number of experiments to analyze the performance of the program. Shen et al. (2018) presented a scheme that can not only protect sensitive information and release other information, so that data can be shared under the premise of protecting sensitive information, but also can effectively carry out remote data integrity audit. Li et al. (2018) proposed an integrity

checking technology, which is mainly for group shared data. This scheme reduces the cost of computation and communication by aggregating tabs in the process of data verification, and supports public verification and effective user revocation, but cannot resist signature forgery attacks. Besides, Zhang et al. (2019) proposed an originality certificateless signature strategy without trusted center, and the key exchange is utilized during the production process of confidential key, which can eliminate the security channel to achieve system robustness.

1.2 Our contributions

Our scheme is primarily aimed at solving the fundamental and challenging problems in federated training neural networks, including confidentiality of local gradients, verifiability of global parameters and untrustworthiness of the of third-party entity. The main contributions are summarized as follows:

- We put forward a verifiable federated training scheme that supports data privacy protection over deep neural networks. Our scheme is based on the framework of VerifyNet.
- So as to ensure the confidentiality of the user's local gradients, a double-masking protocol is used in our scheme. In our scheme, aggregation technology of data tags is used to ensure that the information returned by the CS can be verified. In order to be more suitable for real scenarios, we remove the trusted center in our scheme, assuming that the key generation center (KGC) is veracious but inquisitive.
- The proposed scheme is proven secure, which can resist forgery attacks of tag and verification proof. Furthermore, performance assessment shows that our scheme is efficient.

1.3 Organization

The rest of the paper is organized as below. Some of the relevant preliminaries are described in Sect. 2. In Sect. 3, we introduce the system model, relevant definition of algorithm and security model. We explicitly put forward our scheme in Sect. 4. Then we carry out the security analysis in Sect. 5, and conduct the performance analysis in Sect. 6. We summarize our scheme in Sect. 7.

2 Preliminaries

2.1 Bilinear map

Bilinear mapping is a commonly used mathematical foundation for cryptography, and its related knowledge is introduced as follows. Group G_1, which has a generator g, and group G_2 are two multiplicative cyclic groups. And the order of these two groups is prime q. There is a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$, which meets three characteristics (Li et al. 2018).

- Bilinearity. For all $a, b \in Z_q^*$, $g_1, g_2 \in G_1$, there is $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$.
- Non-degeneracy. Existing $g_1, g_2 \in G_1$, with $\hat{e}(g_1, g_2) \neq 1$.
- Computability. For all $g_1, g_2 \in G_1$, an efficient algorithm exists to calculate $\hat{e}(g_1, g_2)$.

2.2 Computational Diffie-Hellman (CDH) problem

Preset g, g^a and g^b, for undiscovered $a, b \in Z_q^*$, calculate g^{ab}. If the CDH problem in G_1 is impracticable in computation, then the CDH hypothesis in G_1 holds (Shen et al. 2018). We define ε as a negligible value and $Adv_{G_1,\mathcal{A}}^{CDH}$ as the advantage of $\mathcal{A}$ to overcome the CDH hard problem in G_1 (Li et al. 2018), where

$$Adv_{G_1,\mathcal{A}}^{CDH} = Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \leftarrow Z_q^*] \leq \varepsilon.$$

2.3 Key agreement

Diffie and Hellman gave the definition of public key cryptography without precedent and proposed the Diffie-Hellman key exchange algorithm (Diffie and Hellman 1976), which works out the key distribution problem in the symmetric cryptosystem, so that the communication parties can exchange the shared key securely through the open channel. Its security is relies on the strength of the difficulty of the discrete logarithm problem. This protocol is used in our verifiable system. A group G_1 is given, which has a is a generator g. The private/public key of u_i is put up as $KA.gen(G_1, g, q) \rightarrow (sk_i, g^{sk_i})$, where q is the prime order of G_1. Furthermore, assuming that the private/public key of u_j is (sk_i, g^{sk_i}). The negotiation key can be generated by algorithm $KA.agree(sk_i, g^{sk_j}) \rightarrow s_{i,j}$ for these two users.

2.4 Secret sharing

The idea of secret sharing is to split the secrets in a suitable methods. It make further put each share under different user's custody after splitting in the system. A user cannot restructure the secret information. No other than a few users work together to restructure the secret message. More importantly, secrets can still be fully restructured when problems occur with any of the corresponding range of users. The Shamir Secret Sharing was proposed by Shamir based on Lagrange interpolation and vector methods (Shamir 1979). This protocol Shamir (1979) is applied to our verifiable

system. Define the user set $U, |U| = N$. Define t as threshold value and x as the secret. The share of every user in U is created as $\mathsf{S.share}(x, t, U) \rightarrow \{(u_i, x_i)\}_{u_i \in U}$. The secret will be capable of recovering $S.recovn(\{x_i\}_{u_i \in U_1}, t) \rightarrow x$, where $U_1 \subseteq U$ and $|U_1| \geq t$.

2.5 Symmetric encryption

Symmetric encryption algorithm is also called traditional encryption algorithm. Due to its high efficiency, generally, this encryption operation is adopted when the sender needs to encrypt massive data. Symmetric encryption is also called key encryption. The encryption key and decryption key should be the identical in a symmetric encryption system. Given the key sk and the information x to be encrypted, the encrypted information is obtained by the algorithm $\mathsf{AE.enc}(sk, x) \rightarrow \chi$. χ can be decrypted by the algorithm $\mathsf{AE.dec}(sk, \chi) \rightarrow x$.

3 Model and definition

3.1 System model

In Fig. 1 we can see that the system model involves three types of entities: the users, the KGC and the CS. In this system, the CS mainly generates the global parameters and transmits them to all users. The KGC is primarily held responsible for generating user keys. The user sends encrypted local gradients to CS and accepts the global parameter generated by the CS.

During the federated learning, three issues need to be considered. Firstly, in order to take precautions against the adversary obtaining the sensitive information of the user maliciously, it is necessary to keep the local gradients of the user confidential. Secondly, it is necessary for the user to inspect whether the result returned by the CS is forged. Thirdly, it is necessary to remove the trusted third-party entity because it is not practical in real system.

3.2 Definition of algorithms

We briefly introduce the definition of the algorithm involved in the scheme.

- Initialization. The Setup algorithm carries in the security parameter, bulks out the master secret key s and public parameter *params*.
- Key Generation. It inputs public parameter, outputs the user key pair $\{(x_i, D_i), (X_i, \Phi_i)\}$.
- Key Sharing. It inputs the private key of u_i and the public key of u_j, outputs the shares of u_i to u_j.

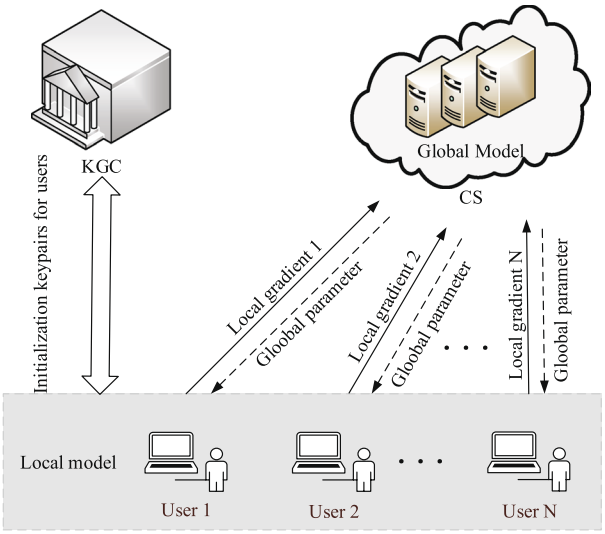

Fig. 1 The system model.

- Masked Input. It inputs the keys of users and local gradient m_i, outputs the encrypted local gradients $\hat{m}_i$.
- TagGen. It inputs the private key of u_i and the local gradient m_i, outputs tag σ_i of the local gradient.
- Unmasking. It inputs the encrypted local gradient $\hat{m}_i$, outputs the aggregated parameter m.
- Proof Generation: It inputs the encrypted local gradient $\hat{m}_i$ and the tag σ_i of the local gradient m_i, outputs the proof information for validation σ.
- Verification. It inputs the information for validation σ and the public keys of users, outputs the verification result.

3.3 Security model

Our scheme takes advantage of the idea of key generation in certificateless signatures (Zhang et al. 2019). So three types of adversaries are taken into account (Li et al. 2018), named as $\mathcal{A}_I$, $\mathcal{A}_{II}$, $\mathcal{A}_{III}$, respectively. They have different capabilities. A public key replacement attack can be performed by $\mathcal{A}_I$ even without accessing the master key. $\mathcal{A}_{II}$ can not mount public key replacement attacks but can get the master secret key. $\mathcal{A}_{III}$ cheats the verifier by forging the validation message to be validated. We describe secure performance of our scheme by defining several security games. Besides, a challenger $\mathcal{C}$ is introduced in three games.

Game 1: This game includes the following phases.

- Setup. The challenger $\mathcal{C}$ generates the master secret key s and other parameters by executing the Initialization algorithm. Afterward, it maintains s securely, while sends public parameters to $\mathcal{A}_I$. This is the initialization phase.
- Queries. The $\mathcal{A}_I$ carries out diverse queries to $\mathcal{C}$.

  - *Hash Query.* $\mathcal{A}_I$ makes the query to $\mathcal{C}$, $\mathcal{C}$ obtains the relevant value and sends it to $\mathcal{A}_I$.
  - *Secret Value Query.* For u_i, $\mathcal{C}$ obtains the secret value x_i of u_i, and then sends it to $\mathcal{A}_I$.
  - *Partial Key Query.* For u_i, $\mathcal{C}$ gets the partial key k_i of u_i, and then sends it to $\mathcal{A}_I$.
  - *Full Secret Key Query.* For u_i, $\mathcal{C}$ sends private key (x_i, D_i) of u_i to $\mathcal{A}_I$.
  - *Full Public Key Query.* For u_i, $\mathcal{C}$ sends public key $(X_i, \boldsymbol{\Phi}_i)$ of u_i to $\mathcal{A}_I$.
  - *Public Key Replacement Query.* $\mathcal{A}_I$ can substitute the public key of the u_i with $(X_i', \boldsymbol{\Phi}_i')$.
  - *Tag Query.* $\mathcal{A}_I$ carries out a query on (u_i, m_i) to $\mathcal{C}$. $\mathcal{C}$ executes the Tag Generation algorithm and returns the tag of m_i to $\mathcal{A}_I$.

- Forgery. For u_i^*, which has the public key $(X_i^*, \boldsymbol{\Phi}_i^*)$, $\mathcal{A}_I$ generates a forged tag σ^* of the m_i^*. When and only when the following essential factors are contented, it will win the game.

  - Firstly, $\mathcal{A}_I$ does not query the full secret key and the partial key of the u_i^*.
  - Secondly, $\mathcal{A}_I$ replaces the public key of the user, who has an identity u_i^*.
  - Thirdly, $\mathcal{A}_I$ does not query the tag of the m_i^*.

Definition 1 If $\mathcal{A}_I$ has an advantage ε to win over the Game 1, where ε is negligible, it can not forge the tag.

Game 2: This game includes the following phases.

- Setup. The challenger $\mathcal{C}$ computes the master secret key s and other parameters by implementing the Initialization algorithm. After that, it sends s and other parameters to $\mathcal{A}_{II}$.
- Queries. The $\mathcal{A}_{II}$ makes different queries to $\mathcal{C}$.

  - *Hash Query.* $\mathcal{A}_{II}$ makes the query to $\mathcal{C}$, $\mathcal{C}$ replies as below. $\mathcal{C}$ obtains the relevant hash value and sends it to $\mathcal{A}_{II}$.
  - *Secret Value Query.* For u_i, $\mathcal{C}$ obtains the secret value x_i and sends x_i to $\mathcal{A}_{II}$.
  - *Full Public Key Query.* For u_i, $\mathcal{C}$ obtains the public key $(X_i, \boldsymbol{\Phi}_i)$ and sends $(X_i, \boldsymbol{\Phi}_i)$ to $\mathcal{A}_{II}$.
  - *Tag Query.* $\mathcal{A}_{II}$ carries out the query on (u_i, m_i) to $\mathcal{C}$. $\mathcal{C}$ executes the Tag Generation algorithm and returns the tag of m_i to $\mathcal{A}_{II}$.

- Forgery. For u_i^*, $\mathcal{A}_{II}$ generates a forged tag σ^* of the m_i^*. When the following essential conditions are fulfilled, it will win the game.

  - $\mathcal{A}_{II}$ does not query the secret value of the u_i^*.
  - $\mathcal{A}_{II}$ does not query the tag of the m_i^*.

Definition 2 If $\mathcal{A}_{II}$ has an advantage ε to win the Game 2, where ε is negligible, it can not forge the tag.

Game 3: $\mathcal{A}_{III}$ tries to spoof the proof of verification result to trick the verifier. The game contains the following phases.

- Setup. The challenger $\mathcal{C}$ computes the master key, the public parameters and keys of all user. After that, the private keys and the master key are kept securely. While the public parameters is sent to $\mathcal{A}_{III}$.
- Tag Query. $\mathcal{A}_{III}$ carries out a query on (u_i, m_i) to $\mathcal{C}$. $\mathcal{C}$ carries out the Tag Generation algorithm aiming at getting the tag of the m_i, and sends the tag to $\mathcal{A}_{III}$.
- Forge. $\mathcal{A}_{III}$ forges the proof of verification result and returns it to $\mathcal{C}$. $\mathcal{A}_{III}$ wins the game when the information is verified, otherwise, the forgery fails.

Definition 3 If $\mathcal{A}_{III}$ has an advantage ε to win the Game 3, where ε is negligible, it can not forge the proof of verification result.

4 The proposed scheme

4.1 A detailed description of our scheme

We specifically illustrate our scheme in this section. We define a user set U with $|U| = N$. Assume that the identities of all users in this system are orderly, and all users can send messages in time. Figure 2 mainly shows our algorithm flow based on VerifyNet.

- Initialization: Let G_1 and G_2 be two multiplicative cyclic groups of prime order q. There exits a bilinear map $\hat{e} : G_1 \times G_1 \to G_2$. g is a generator of G_1. $H_1 : Z_q^* \to G_1, H_2 : G_1 \to Z_q^*, H_3 : Z_q^* \times G_1 \times G_1 \to Z_q^*$ and $H_4 : Z_q^* \to Z_q^*$ are four general secure hash functions. $H_5 : Z_q^* \to G_1$ is a secure homomorphic hash function. $\mathsf{PRG}(s_{i,j})$ is defined as a pseudo-random generator with seed $s_{i,j}$. The KGC randomly randomly selects two numbers $\psi_1 \in Z_q^*$ and $\psi_2 \in Z_q^*$, and then computes $K_1 = H_1(\psi_1)$ and $K_2 = H_1(\psi_2)$. The KGC randomly picks $s \in Z_q^*$ and generates P by calculating $P = g^s$. Here s is regarded as the master key. It keeps s private and publishes $params = (q, g, G_1, G_2, \hat{e}, P, H_1, H_2, H_3, H_4, H_5, K_1, K_2)$.
- Key Generation: The algorithm mainly includes the following three sub algorithms.

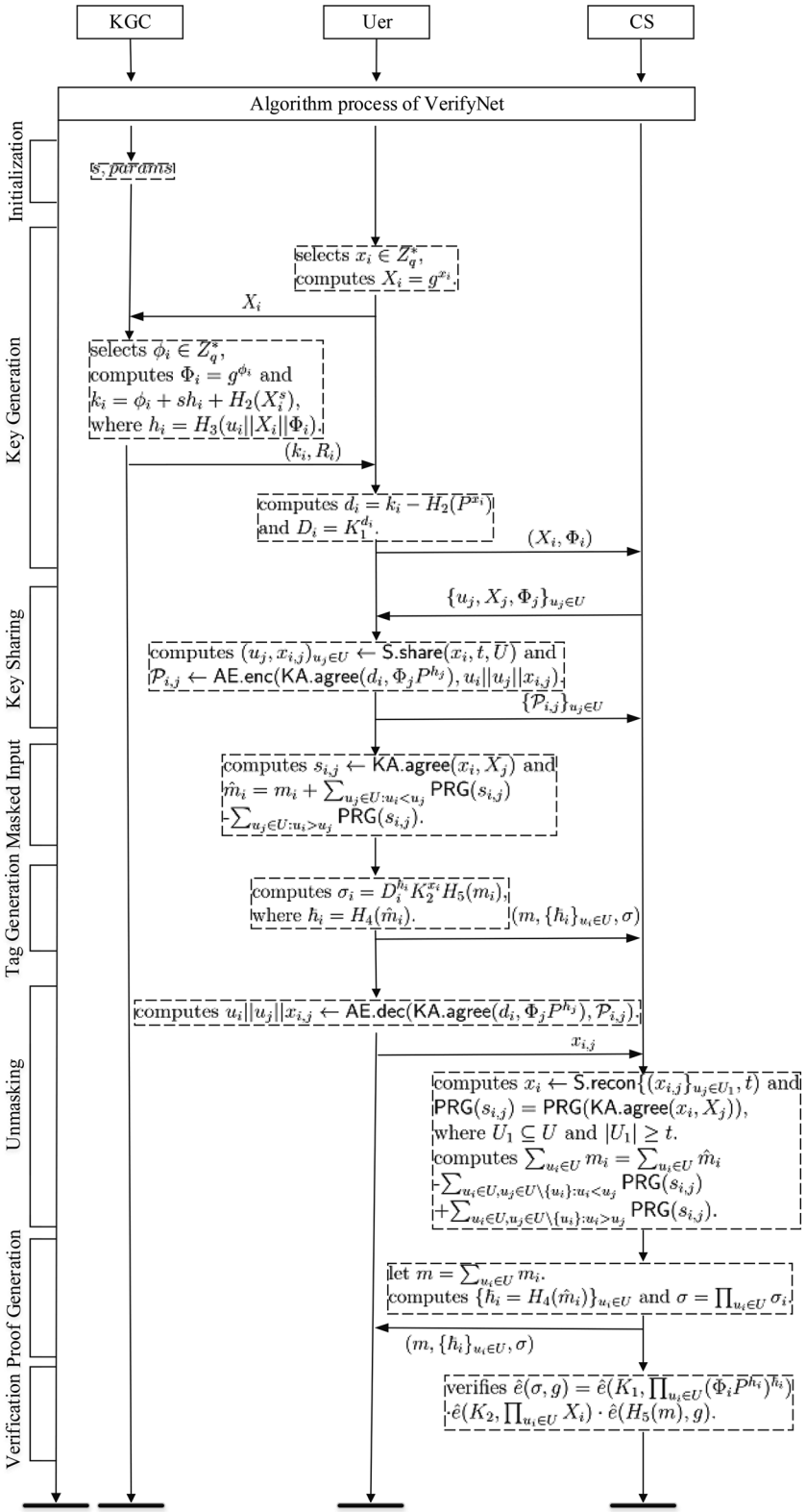

Fig. 2 Algorithm flow of the proposed scheme

– Secret Value Generation: A user u_i, which possess the identity u_i, $1 \leq i \leq N$, first selects $x_i \in Z_q^*$, then calculates $X_i = g^{x_i}$. After that u_i sends X_i to the KGC.

– Partial Key Generation: After receives X_i, the KGC picks $\phi_i \in Z_q^*$, then calculates $\boldsymbol{\Phi}_i = g^{\phi_i}$ and $k_i = \phi_i + sh_i + H_2(X_i^s)$, where $h_i = H_3(u_i||X_i||\boldsymbol{\Phi}_i)$.

The KGC subsequently returns $(k_i, \boldsymbol{\Phi}_i)$ to u_i, which put to use public channels.

- Full Key Generation: After receives $(k_i, \boldsymbol{\Phi}_i)$, u_i firstly computes $d_i = k_i - H_2(P^{x_i})$. Then u_i checks whether $g^{d_i} = \boldsymbol{\Phi}_i \cdot P^{H_3(u_i||X_i||\boldsymbol{\Phi}_i)}$ holds. If not, aborts and starts over. Then the user u_i computes $D_i = K_1^{d_i}$. (x_i, D_i) and $(X_i, \boldsymbol{\Phi}_i)$ will be applied to encrypt and decrypt the local gradients m_i of u_i. Finally the u_i returns $(X_i, \boldsymbol{\Phi}_i)$ to the CS.

- Key Sharing: When receives the information from all users in the system, the CS broadcasts $\{u_j, X_j, \boldsymbol{\Phi}_j\}_{u_j \in U}$ to each user. After receiving the information from the CS, the user u_i checks whether all of these key pairs $(X_j, \boldsymbol{\Phi}_j)$ are distinctive. If not, aborts and starts over. Then the user u_i computes the shares of u_i to u_j as $(u_j, x_{i,j})_{u_j \in U} \leftarrow \mathsf{S.share}(x_i, t, U)$ and $\mathcal{P}_{i,j} \leftarrow \mathsf{AE.enc}(\mathsf{KA.agree}(d_i, \boldsymbol{\Phi}_j P^{h_j}), u_i||u_j||x_{i,j})$. After that, the user u_i returns $\{\mathcal{P}_{i,j}\}_{u_j \in U}$ to the CS.
- Masked Input: When receives the information from all users in the system, the user u_i computes shared key with every user $u_j \in U \setminus \{u_i\}$ in the system $s_{i,j} \leftarrow \mathsf{KA.agree}(x_i, X_j)$. After then the user u_i encrypts the local gradient m_i as $\hat{m}_i = m_i + \sum_{u_j \in U: u_i < u_j} \mathsf{PRG}(s_{i,j}) - \sum_{u_j \in U: u_i > u_j} \mathsf{PRG}(s_{i,j})$.
- Tag Generation: So as to inspect the correctness of the returned results from CS, u_i computes the tag of the local gradient $\sigma_i = D_i^{\hbar_i} K_2^{x_i} H_5(m_i)$, where $\hbar_i = H_4(\hat{m}_i)$. After that, u_i sends $(\hat{m}_i, \hbar_i, \sigma_i)$ to the CS.
- Unmasking: During the unmasking procedure, the user u_i, firstly, calculates $u_i||u_j||x_{i,j} \leftarrow \mathsf{AE.dec}(\mathsf{KA.agree}(d_i, \boldsymbol{\Phi}_j P^{h_j}), \mathcal{P}_{i,j})$ and returns $x_{i,j}$ to the CS. Since gets the $x_{i,j}$ from all users, the CS computes $x_i \leftarrow \mathsf{S.recon}\{(x_{i,j}\}_{u_j \in U_1}, t)$ and $\mathsf{PRG}(s_{i,j}) = \mathsf{PRG}(\mathsf{KA.agree}(x_i, X_j))$, where $U_1 \subseteq U$ and $|U_1| \geq t$. After that, the CS computes aggregated parameters for all users as $\sum_{u_i \in U} m_i = \sum_{u_i \in U} \hat{m}_i - \sum_{u_i \in U, u_j \in U \setminus \{u_i\}: u_i < u_j} \mathsf{PRG}(s_{i,j}))$ $+ \sum_{u_i \in U, u_j \in U \setminus \{u_i\}: u_i > u_j} \mathsf{PRG}(s_{i,j})$.
- Proof Generation: let $m = \sum_{u_j \in U} m_i$, the CS calculates $\{\hbar_i = H_4(\hat{m}_i)\}_{u_i \in U}$ and $\sigma = \prod_{u_i \in U} \sigma_i$ and broadcasts $(m, \{\hbar_i\}_{u_i \in U}, \sigma)$ to each user.
- Verification: Firstly, u_i needs to verify whether the equation $\hat{e}(\sigma, g) = \hat{e}(K_1, \prod_{u_i \in U} (\boldsymbol{\Phi}_i P^{h_i})^{\hbar_i}) \cdot \hat{e}(K_2, \prod_{u_i \in U} X_i) \cdot \hat{e}(H_5(m), g)$ holds or not. If holds, the user accepts the global parameter m. The correctness of the equation indicated below:

$$
\begin{aligned}
\hat{e}(\sigma, g) &= \hat{e}(\prod_{u_i \in U} \sigma_i, g) \\
&= \prod_{u_i \in U} \hat{e}(D_i^{\hbar_i} K_2^{x_i} H_5(m_i), g) \\
&= \prod_{u_i \in U} (\hat{e}(D_i^{\hbar_i}, g) \cdot \hat{e}(K_2^{x_i}, g) \cdot \hat{e}(H_5(m_i), g)) \\
&= \hat{e}(\prod_{u_i \in U} K_1^{(sh_i + \phi_i)\hbar_i}, g) \cdot \hat{e}(\prod_{u_i \in U} K_2^{x_i}, g) \\
&\cdot \hat{e}(\prod_{u_i \in U} H_5(m_i), g) \\
&= \hat{e}(K_1, \prod_{u_i \in U} g^{(sh_i + \phi_i)\hbar_i}) \cdot \hat{e}(K_2, \prod_{u_i \in U} g^{x_i}) \\
&\cdot \hat{e}(H_5(m), g) \\
&= \hat{e}(K_1, \prod_{u_i \in U} (\boldsymbol{\Phi}_i P^{h_i})^{\hbar_i}) \cdot \hat{e}(K_2, \prod_{u_i \in U} X_i) \\
&\cdot \hat{e}(H_5(m), g)
\end{aligned}
$$

5 Security analysis

This segment mainly analyzes the security of our verifiable federated learning scheme.

Theorem 1 *For $\mathcal{A}_I$, the proposed scheme is existentially unforgeable against chosen-message attacks under the CDH assumption.*

Proof Suppose $\mathcal{A}_I$ can triumph the Game 1, which has probability ε, where ε is non-negligible. Given an instance (g, g^a, g^b) of the CDH problem, a simulator $\mathcal{B}$ can find a solution g^{ab} with a probability ε' by interacting with $\mathcal{A}_I$, where ε' is non-negligible.

- Setup. The simulator $\mathcal{B}$ chooses the master secret key $a \in Z_q^*$ which is kept securely, sets $P = g^a$. Then $\mathcal{B}$ selects two values $\psi_1, \psi_2 \in Z_q^*$, and produces other public parameters by performing the Initialization algorithm.
- Queries. The $\mathcal{A}_I$ makes the following queries.
  - *Hash-1 Query.* $\mathcal{A}_I$ carries out the query for (ψ_1, ψ_2). In the event that (ψ_1, ψ_2) has been queried, $\mathcal{B}$ returns (τ_1, τ_2) to $\mathcal{A}_I$. If not, $\mathcal{B}$ selects $t_1, t_2 \in Z_q^*$, then calculates $\tau_1 = g^{t_1}$ and $\tau_2 = g^{t_2}$. $\mathcal{B}$ returns (τ_1, τ_2) and retains a list $Ł_{hash1}$ of the format $(\psi_1, \psi_2, t_1, t_2, \tau_1, \tau_2)$.
  - *Secret Value Query.* $\mathcal{A}_I$ carries out Secret Value Query for u_i. In the event that u_i has been queried, $\mathcal{B}$ returns the exiting x_i. Otherwise, $\mathcal{B}$ picks $x_i \in Z_q^*$, calculates $X_i = g^{x_i}$, then returns x_i. $\mathcal{B}$ retains a list $Ł_{usk}$ of the format (u_i, x_i, X_i).

- *Partial Key Query.* $\mathcal{A}_I$ carries out the query for u_i. In the event that u_i has not been queried, $\mathcal{B}$ throws a coin $\pi_i \in \{0,1\}$, where $Pr[\pi_i = 1] = \xi$. When $\pi_i = 0$, $\mathcal{B}$ randomly selects $\phi_i \in Z_q^*$, calculates $\boldsymbol{\Phi}_i = g^{\phi_i}$, $h_i = H_3(u_i||X_i||\boldsymbol{\Phi}_i)$ and $k_i = \phi_i + ah_i + H_2(X_i^a)$, where (u_i, x_i, X_i) is obtained by querying Secret Value Query and h_i can be obtained by querying Hash-3 Query. Then it returns k_i and attaches $(u_i, \phi_i, \boldsymbol{\Phi}_i, k_i, \pi_i)$ into $Ł_{psk}$. When $\pi_i = 1$, $\mathcal{B}$ aborts. When u_i has been queried, $\mathcal{B}$ checks whether π_i is not 1. $\mathcal{B}$ returns k_i if $\pi_i = 0$, and aborts otherwise.
- *Full Secret Key Query.* $\mathcal{A}_I$ carries out the query for u_i. In the event of u_i queried, $\mathcal{B}$ detects whether the last component of the commensurable tuple is 1. In the event that the last component is 1, $\mathcal{B}$ aborts, otherwise returns x_i and D_i. In the event that u_i has not been queried, $\mathcal{B}$ obtains (u_i, x_i, X_i) and $(u_i, \phi_i, \boldsymbol{\Phi}_i, k_i, \pi_i)$ by querying Secret Value Query and Partial Key Query. If $\pi_i = 1$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ calculates $d_i = k_i - H_2(X_i^a)$ and $D_i = \tau_1^{d_i}$, where τ_1 is obtained from $Ł_{hash1}$. Then $\mathcal{B}$ inserts (u_i, x_i, D_i, π_i) into $Ł_{sk}$ and returns x_i and D_i. So far, the public secret key query is completed.
- *Full Public Key Query.* $\mathcal{A}_I$ performs Full Public Key Query for an identity u_i. In the event that u_i has been queried, $\mathcal{B}$ returns X_i and $\boldsymbol{\Phi}_i$. when u_i has not been queried, $\mathcal{B}$ obtains (u_i, x_i, X_i) and $(u_i, \phi_i, \boldsymbol{\Phi}_i, k_i, \pi_i)$ by querying Secret Value Query and Partial Key Query. $\mathcal{B}$ returns X_i and $\boldsymbol{\Phi}_i$ and inserts $(u_i, X_i, \boldsymbol{\Phi}_i, \pi_i)$ into $Ł_{pk}$.
- *Public Key Replacement Query.* $\mathcal{A}_I$ performs the query with $(u_i, X_i', \boldsymbol{\Phi}_i')$. $\mathcal{B}$ refreshes the corresponding tuple in $Ł_{pk}$.
- *Hash-2 Query.* $\mathcal{A}_I$ makes hash function queries for m_i. When m_i has been queried, $\mathcal{B}$ returns g^{w_i} to $\mathcal{A}_I$, or else that $\mathcal{B}$ selects a value $w_i \in Z_q^*$, then computes g^{w_i}. $\mathcal{B}$ sends g^{w_i} to $\mathcal{A}_I$ and attaches (m_i, w_i, g_i^w) into $Ł_{hash2}$.
- *Hash-3 Query.* $\mathcal{A}_I$ makes hash function queries for $u_i||X_i||\boldsymbol{\Phi}_i$. When $u_i||X_i||\boldsymbol{\Phi}_i$ has been queried, $\mathcal{B}$ sends h_i to $\mathcal{A}_I$, or else that $\mathcal{B}$ randomly selects $h_{1,i} \in Z_q^*$ and calculates $h_i = bh_{1,i}$. $\mathcal{B}$ returns h_i and retains a list $Ł_{hash3}$ of the format $(u_i||X_i||\boldsymbol{\Phi}_i, h_i)$.
- *Hash-4 Query.* $\mathcal{A}_I$ makes hash function queries for $\hat{m}_i$. When $\hat{m}_i$ has been queried, $\mathcal{B}$ sends $\hbar_i$ to $\mathcal{A}_I$, or else that $\mathcal{B}$ selects a value $\hbar_i \in Z_q^*$, sends it to $\mathcal{A}_I$ and attaches $(\hat{m}_i, \hbar_i)$ into $Ł_{hash4}$.
- *Tag Query.* $\mathcal{A}_I$ carries out the query with (u_i, m_i). $\mathcal{B}$ checks whether π_i is 0. If $\pi_i' = 1$, $\mathcal{B}$ aborts. When $\pi_i' = 0$, $\mathcal{B}$ gets $(u_i, x_i, X_i, \phi_i', \boldsymbol{\Phi}_i', k_i', \pi_i')$ from $Ł_{sk}$, $H_1(\psi_2)$ from $Ł_{hash1}$, $H_5(m_i)$ from $Ł_{hash2}$ and $H_4(\hat{m}_i)$ from $Ł_{hash4}$. $\mathcal{B}$ generates a valid tag by executes the Tag Generation algorithm and returns it to $\mathcal{A}_I$.
- Forgery. $\mathcal{A}_I$ outputs $(u_i^*, \hat{m}_i^*, m_i^*, X_i^*, \boldsymbol{\Phi}_i^*.\sigma_i^*)$, σ_i^* is the forged tag of the local gradient m_i^* by substituting the public key of u_i^*.
- Analysis. If $\mathcal{A}_I$ wants to win the Game 1, $\mathcal{B}$ can do the following. $\mathcal{B}$ firstly retrieves $(u_i, X_i, \boldsymbol{\Phi}_i, \pi_i)$ from $Ł_{pk}$. If $\pi_i^* = 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ computes $H_5(m_i^*) = g^{w_i}$ and retrieves $H_1(\psi_1) = g^{t_1}$ and $H_1(\psi_2) = g^{t_2}$ from $Ł_{hash1}$, $H_4(\hat{m}_i^*) = \hbar_i^*$ from $Ł_{hash4}$ and $h_i = bh_{1,i}$ from $Ł_{hash3}$. $\mathcal{B}$ can get $g^{ab} = (\frac{\sigma_i^*}{\boldsymbol{\Phi}_i^{t_1\hbar_i^*} X_i^{t_2} g^{w_i}})^{\frac{1}{t_1\hbar_i^* h_{1,i}}}$. $\mathcal{B}$ exports the correct value g^{ab} with the probability $\varepsilon' \geq \varepsilon \cdot \xi \cdot (1-\xi)^{q_{psk}+q_{sk}+q_T} \geq \varepsilon/((q_{psk}+q_{sk}+q_T)\cdot e)$, where q_{psk}, q_{sk} and q_T represent the times of Partial Key Query, Full Secret Key Query and Tag Query respectively. In this situation, $\mathcal{B}$ solves the CDH problem, obviously this is impossible according to CDH assumption.

Theorem 2 *For $\mathcal{A}_{II}$, the proposed scheme is existentially unforgeable against chosen-message attacks under the CDH assumption.*

Proof Suppose $\mathcal{A}_{II}$ can triumph the Game 2, which has probability ε, where ε is also non-negligible. Given an instance (g, g^a, g^b) of the CDH problem, a simulator $\mathcal{B}$ can find a solution g^{ab} with a probability ε' by interacting with $\mathcal{A}_{II}$, where ε' is also non-negligible.

- Setup. The simulator $\mathcal{B}$ chooses the master key $s \in Z_q^*$, then selects two numbers $\psi_1 \in Z_q^*$ and $\psi_2 \in Z_q^*$ and creates other public parameters by running the Initialization algorithm. Then s and public parameters are sent to $\mathcal{A}_{II}$.
- Queries. The $\mathcal{A}_{II}$ makes the following queries.
  - *Hash-1 Query.* $\mathcal{A}_{II}$ carries out the query for (ψ_1, ψ_2). In the event that (ψ_1, ψ_2) has been queried, $\mathcal{B}$ returns (τ_1, τ_2) to $\mathcal{A}_{II}$, or else that $\mathcal{B}$ randomly selects $t_1, t_2 \in Z_q^*$ and calculates $\tau_1 = g^{t_1}$ and $\tau_2 = g^{bt_2}$. $\mathcal{B}$ returns (τ_1, τ_2) and retains a list $Ł_{hash1}$ of the format $(\psi_1, \psi_2, t_1, t_2, \tau_1, \tau_2)$.
  - *Secret Value Query.* $\mathcal{A}_{II}$ makes the query for u_i. When u_i has been queried, $\mathcal{B}$ returns the exiting x_i, or else that $\mathcal{B}$ throws a coin $\pi_i \in \{0,1\}$, where $Pr[\pi_i = 1] = \xi$. If $\pi_i = 0$, $\mathcal{B}$ calculates $X_i = g^{x_i}$ and returns x_i. $\mathcal{B}$ retains a list $Ł_{usk}$ of the format (u_i, x_i, X_i, π_i). If $\pi_i = 1$, $\mathcal{B}$ calculates $X_i = g^{ax_i}$ and attaches (u_i, x_i, X_i, π_i) to $Ł_{usk}$ and aborts.
  - *Full Public Key Query.* $\mathcal{A}_{II}$ performs the query for an identity u_i. In the event that u_i has been queried, $\mathcal{B}$ returns X_i and $\boldsymbol{\Phi}_i$. When u_i has not been queried, $\mathcal{B}$ obtains (u_i, x_i, X_i) by querying Secret Value Query. $\mathcal{B}$ selects a value $\phi_i \in Z_q^*$ and calculates $\boldsymbol{\Phi}_i = g^{\phi_i}$, $\mathcal{B}$ returns X_i and $\boldsymbol{\Phi}_i$ and inserts $(u_i, X_i, \boldsymbol{\Phi}_i, \pi_i)$ into $Ł_{pk}$.

- *Hash-2 Query.* $\mathcal{A}_{II}$ makes hash function queries for m_i. When m_i has been queried, $\mathcal{B}$ returns g^{w_i} to $\mathcal{A}_{II}$, or else that $\mathcal{B}$ picks a value $w_i \in Z_q^*$, then calculates g^{w_i}. $\mathcal{B}$ sends g^{w_i} to $\mathcal{A}_{II}$ and attaches (m_i, w_i, g^{w_i}) into $Ł_{hash2}$.
- *Hash-3 Query.* $\mathcal{A}_{II}$ carries out hash function queries for $u_i||X_i||\boldsymbol{\Phi}_i$. In the event that $u_i||X_i||\boldsymbol{\Phi}_i$ has been queried, $\mathcal{B}$ returns h_i to $\mathcal{A}_{II}$. Otherwise, $\mathcal{B}$ selects $h_{1,i} \in Z_q^*$. Let $h_i = h_{1,i}$, $\mathcal{B}$ returns h_i and retains a list $Ł_{hash3}$ of the format $(u_i||X_i||\boldsymbol{\Phi}_i, h_i)$.
- *Hash-4 Query.* $\mathcal{A}_{II}$ makes hash function queries for $\hat{m}_i$. When $\hat{m}_i$ has been queried, $\mathcal{B}$ sends $\hbar_i$ to $\mathcal{A}_{II}$, or else that $\mathcal{B}$ picks a value $\hbar_i \in Z_q^*$, sends it to $\mathcal{A}_{II}$ and attaches $(\hat{m}_i, \hbar_i)$ into $Ł_{hash4}$.
- *Tag Query.* $\mathcal{A}_{II}$ makes the query with (u_i, m_i). $\mathcal{B}$ checks whether π_i in the list $Ł_{usk}$ is 0. when $\pi_i' = 1$, $\mathcal{B}$ aborts. when $\pi_i' = 0$, $\mathcal{B}$ gets $H_5(m_i)$ from $Ł_{hash2}$ and h_i from $Ł_{hash3}$ and computes D_i. $\mathcal{B}$ generates a valid tag by executes the Tag Generation algorithm and sends it to $\mathcal{A}_{II}$.

- Forgery. $\mathcal{A}_{II}$ outputs $(u_i^*, \hat{m}_i^*, m_i^*, \sigma_i^*)$, σ_i^* is the forged tag of the local gradient m_i^*.
- Analysis. If $\mathcal{A}_{II}$ wants to win the Game 2, $\mathcal{B}$ can do the following. $\mathcal{B}$ firstly retrieves $(u_i, X_i, \boldsymbol{\Phi}_i, \pi_i)$ from $Ł_{pk}$. In the event of $\pi_i^* = 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ computes $H_5(m_i^*) = g^{w_i^*}$ and retrieves $H_1(\psi_1) = g^{t_1}$ and $H_1(\psi_2) = g^{bt_2}$ from $Ł_{hash1}$, $H_4(\hat{m}_i^*) = \hbar_i^*$ from $Ł_{hash4}$ and $h_i = h_{1,i}$ from $Ł_{hash3}$. In addition, $\mathcal{B}$ obtains $X_i = g^{ax_i^*}$. $\mathcal{B}$ can get $g^{ab} = (\frac{\sigma_i^*}{g^{(\phi_i^* + sh_i)r_1 \hbar_i^* + w_i}})^{\frac{1}{t_2 x_i^*}}$. $\mathcal{B}$ exports the correct value g^{ab}, which has the probability $\varepsilon' \geq \varepsilon \cdot \xi \cdot (1-\xi)^{q_{usk}+q_T} \geq \varepsilon/((q_{usk}+q_T)\cdot e)$, where q_{usk} and q_T represent the times of Secret Value Query and Tag Query respectively. $\mathcal{B}$ solves the CDH problem. Obviously, this is impossible according to CDH assumption.

Theorem 3 *$\mathcal{A}_{III}$ has an advantage ε to win the Game* 3, *where ε is negligible.*

Proof Assume that the proof used for verification is $(m, \{\hbar_i\}_{u_i \in U}, \sigma)$. If $\mathcal{A}_{III}$ wins the Game 3, then $\mathcal{A}_{III}$ generates a proof (m, σ) and passes the validation equation $\hat{e}(\sigma, g) = \hat{e}(K_1, \prod_{u_i \in U}(\boldsymbol{\Phi}_i P^{h_i})^{\hbar_i}) \cdot \hat{e}(K_2, \prod_{u_i \in U} X_i) \cdot \hat{e}(H_5(m), g)$ Since $m = \sum_{u_i \in U} m_i$, At least one parameter $m_i \neq m_i^*$ exists. According to the collision resistance of hash function, $H_5(m_i) \neq H_5(m_i^*)$. It can be visible from this that $\prod_{u_j \in U} H_5(m_j) \neq \prod_{u_j \in U \setminus \{u_i\}} H_5(m_j) H_5(m_i^*)$. It is further known that $H_5(m) \neq H_5(m^*)$. From the collision resistance of hash function, it is impossible to forge the proof of verification result. □

6 Performance analysis

As demonstrated before, the scheme VerifyNet Guowen et al. (2020) has the limitation of relying on a fully trusted third party. Additionally, the performance of ensuring the correctness of returned results remains to be improved. In our scheme, we address both security and performance issues in VerifyNet based on the techniques of data auditing and certificateless tag generation. To show the advantages of the proposed scheme, we compare our scheme with schemes Li et al. (2018) and Shen et al. (2018). The associated notations are shown in Table 1.

We mainly consider two phases of computational overhead, including the efficiency of tag generation and the efficiency of validation. M_{G_1} and E_{G_1} respectively represent multiplication operation and exponentiation operation in G_1. M_{G_2} and E_{G_2} respectively represent the multiplication operation and the exponentiation operation in G_2. *Pair* represents the pairing operation. In the stage of proof generation and verification, the main calculations include that the CS needs to generate proof σ and that the user needs to verify the proof, so the computation cost of the CS is $(N-1)M_{G_1}$ and the computation cost of the user is $(3N-2)M_{G_1} + 2NE_{G_1} + 2M_{G_2} + 4Pair$ in our scheme. In addition, we display the computation cost in schemes Li et al. (2018) and Shen et al. (2018) as shown in Table 2, in which d is the size of the subset group and sets to N.

We compare the communication cost of our scheme with schemes Li et al. (2018) and Shen et al. (2018). Assume that the number of local gradients to be verified is the same as the number of users. In the proof generation stage, the CS returns the proof $(m, \{\hbar_i\}_{u_i \in U}, \sigma)$ to the user, and the communication overhead of the CS is $(N+1)|q| + |p|$, as shown in Table 3. In addition, we display the communication cost in schemes Li et al. (2018) and Shen et al. (2018).

Our experiment was carried out under the windows platform and used a java programming language with Java Pairing Based Cryptography. So as to facilitate comparison, we adopt the same parameters as scheme Shen et al. (2018), that is, $|q|$ and $|p|$ is set to 160 bits and 512 bits respectively.

The efficiency of the proof generation and verification can be shown in Fig. 3. In the case, when the number of local gradients grows from 100 to 500, the time it takes to

Table 1 Notations

Notation	Description
N	Number of users
l	Length of user identity
$\|q\|$	The size of an element on Z_q^*
$\|p\|$	The size of an element on G_1

Table 2 Comparisons of computation overhead

Scheme	Proof generation	Verification
Li et al. (2018)	$(N-d)M_{G_1} + NE_{G_1}$	$(N+2d)M_{G_1} + (N+d)E_{G_1} + dM_{G_2} + (d+2)Pair$
Shen et al. (2018)	$(N-1)M_{G_1} + NE_{G_1}$	$(N+l)M_{G_1} + (N+l+1)E_{G_1} + 2M_{G_2} + 2E_{G_2} + 4Pair$
Ours	$(N-1)M_{G_1}$	$(3N-2)M_{G_1} + 2NE_{G_1} + 2M_{G_2} + 4Pair$

Table 3 Comparisons of communication overhead

Scheme	Proof verification
Li et al. (2018)	$(d+3)\|q\| + d\|p\|$
Shen et al. (2018)	$(2N+1)\|q\| + \|p\|$
Ours	$(N+1)\|q\| + \|p\|$

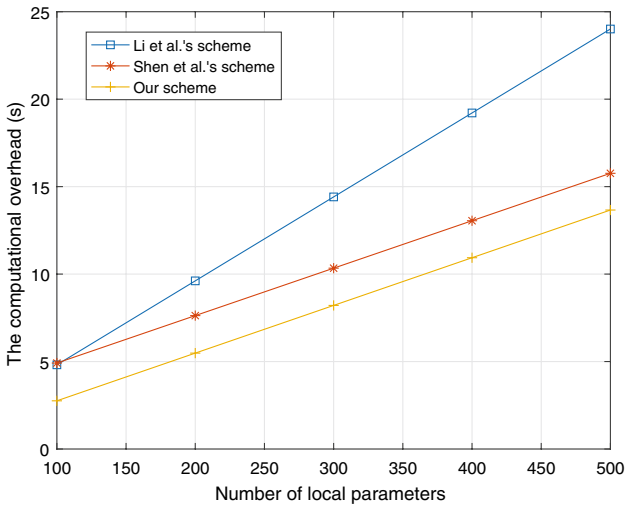

Fig. 3 Comparison of the computation overhead of proof generation and verification

generate proof and verify with our scheme is from 2.7556s to 13.6620s. In our scheme, the computational cost of proof generation and verification increases linearly with the number of local gradients. And our scheme takes less time than schemes Li et al. (2018) and Shen et al. (2018). In the proof generation and verification, the communication cost is shown in Fig. 4. The size of parameter is set 160 bits. In our scheme and schemes Li et al. (2018) and Shen et al. (2018), the communication cost is growing as the number of parameters increases. Obviously, our scheme is more effective than schemes Li et al. (2018) and Shen et al. (2018).

7 Conclusion

In this paper, we come up with a verifiable federated learning scheme over deep neural networks, which supports user's privacy protection. The idea of the key exchange is

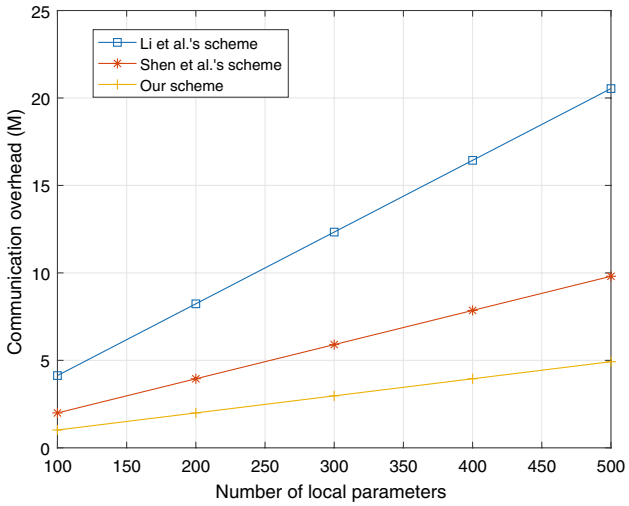

Fig. 4 Comparison of the communication overhead of proof generation and verification

introduced to generate the partial key of the user. Besides, the double masking protocol and tag aggregation method are used to achieve users' privacy protection and verify the correctness of the results returned by the server. Security analysis and efficiency analysis indicate that our scheme is secure and efficient.

Acknowledgements This research is supported by the Innovation Capability Support Program of Shaanxi (Grant No. 2020KJXX-052), the Shaanxi Special Support Program Youth Top-notch Talent Program, the Key Research and Development Program of Shaanxi (Grant No. 2019KW-053, 2020ZDLGY08-04), and the Natural Science Basic Research Plan in Shaanxi Province of China (Grant No. 2019JQ-866).

References

Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS'17, pp 1175–1191, New York, NY, USA. Association for Computing Machinery

Chen Y, Luo F, Li T, Xiang T, Liu Z, Li J (2020) A training-integrity privacy-preserving federated learning scheme with trusted execution environment. Inf Sci 522:69–79

Cong W, Chow Sherman SM, Qian W, Kui R, Wenjing L (2011) Privacy-preserving public auditing for secure cloud storage. IEEE Trans Comput 62(2):362–375

Diffie W, Hellman Martin E (1976) New directions in cryptography. IEEE Trans Inf Theory 22(6):644–654

Dong Z (2019) Federal learning: the second goal of ai security after deep learning.https://www.leiphone.com/news/201911/ziMkFyZXf1ERiniG.html

Fan Y, Lin X, Tan G, Zhang Y, Dong W, Lei J (2019) One secure data integrity verification scheme for cloud storage. Future Gener Comput Syst 96:376–385

García-Gil D, Luque-Sánchez F, Luengo J, García S, Herrera F (2019) From big to smart data: iterative ensemble filter for noise filtering in big data classification. Int J Intell Syst 34(12):3260–3274

Gu X, Angelov PP, Soares EA (2020) A self-adaptive synthetic oversampling technique for imbalanced classification. Int J Intell Syst 35(6):923–943

Guowen X, Li H, Dai Y, Yang K, Lin X (2019) Enabling efficient and geometric range query with access control over encrypted spatial data. IEEE Trans Inf Forensics Secur 14(4):870–885

Guowen X, Li H, Liu S, Yang K, Lin X (2020) Verifynet: Secure and verifiable federated learning. IEEE Trans Inf Forensics Secur 15:911–926

Hitaj B, Ateniese G, Perez-Cruz F (2017) Deep models under the gan: Information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS'17, pp 603–618, New York, NY, USA, Association for Computing Machinery

Jiang T, Chen X, Ma Jianfeng (2015) Public integrity auditing for shared dynamic cloud data with group user revocation. IEEE Trans Comput 65(8):2363–2373

Le Trieu P, Yoshinori A, Takuya H, Lihua W, Shiho M (2018) Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans Inf Forensics Secur 13(5):1333–1345

Li P, Li T, Ye H, Li J, Chen X, Xiang Y (2018) Privacy-preserving machine learning with multiple data providers. Future Gener Comput Syst 87:341–350

Li T, Li J, Chen X, Liu Z, Lou W, Hou T (2020) Npmml: a framework for non-interactive privacy-preserving multi-party machine learning. IEEE Trans Depend Secure Comput

Liu C, Cao Y, Luo Y, Chen G, Vokkarane V, Ma Y (2016) Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In Proceedings of the 14th International Conference on Inclusive Smart Cities and Digital Health—Volume 9677, ICOST 2016, pp 37–48, Springer-Verlag, Berlin, Heidelberg

Li J, Yan H, Zhang Y (2018) Certificateless public integrity checking of group shared data on cloud storage. IEEE Trans Serv Comput

Ma X, Yu H, Wang Y, Wang Y (2015) Large-scale transportation network congestion evolution prediction using deep learning theory. Plos One, p 10

Mohammadi M, Al-Fuqaha A, Sorour S, Guizani M (2018) Deep learning for iot big data and streaming analytics: a survey. IEEE Commun Surv Tutor 20(4):2923–2960

Moriai S (2019) Privacy-preserving deep learning via additively homomorphic encryption. In 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH), pp 198–198,

Nan F, Tian H, Wang T, Cai Y, Chen Y et al (2019) A collusion-resistant public auditing scheme for shared cloud data. Int J Inf Technol Manag 18(2/3):195–212

Shamir A (1979) How to share a secret. IEEE Trans Inf Forensics Secur 22(11):612–613

Shao B, Bian G, Wang Y, Shenghao S, Guo C (2018) Dynamic data integrity auditing method supporting privacy protection in vehicular cloud environment. IEEE Access 6:43785–43797

Shen J, Shen J, Chen X, Huang X, Susilo W (2017) An efficient public auditing protocol with novel dynamic structure for cloud data. IEEE Trans Inf Forensics Secur 12(10):2402–2415

Shen W, Qin J, Jia Y, Hao R, Jiankun H (2018) Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. IEEE Trans Inf Forensics Secur 14(2):331–346

Shen W, Qin J, Yu J, Hao R, Hu J, Ma J (2019) Data integrity auditing without private key storage for secure cloud storage. IEEE Trans Cloud Comput

Shokri R, Shmatikov V (2015) Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS'15, pp 1310–1321, New York, NY, USA, Association for Computing Machinery

Sze V, Chen YH, Yang TJ, Emer JS (2017) Efficient processing of deep neural networks: a tutorial and survey. Proc IEEE 105(12):2295–2329

Tian H, Nan F, Jiang H, Chang C-C, Ning J, Huang Yongfeng (2019) Public auditing for shared cloud data with efficient and secure group management. Inf Sci 472:107–125

Wang J, Chen X, Huang X, You I, Xiang Y (2015) Verifiable auditing for outsourced database in cloud computing. IEEE Trans Comput 64(11):3293–3303

Yan H, Li J, Han J, Zhang Yichen (2016) A novel efficient remote data possession checking protocol in cloud storage. IEEE Trans Inf Forensics Secur 12(1):78–88

Yang Q, Liu Y, Chen T, Tong Y (2019) Federated machine learning: concept and applications. Acm Trans Intell Syst 10(2):121–1219

Yan H, Li J, Zhang Y (2019) Remote data checking with a designated verifier in cloud storage. IEEE Syst J

Yinghui Z, Deng Robert H, Jiangang S, Kan Y, Dong Z (2018) Tkse: trustworthy keyword search over encrypted data with two-side verifiability via blockchain. IEEE Access 6:31077–31087

Yu Y, Au MH, Ateniese G, Huang X, Susilo W, Dai Y, Min G (2016) Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans Inf Forensics Secur 12(4):767–778

Zhang Y, Chunxiang X, Liang X, Li H, Yi M, Zhang X (2017) Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation. IEEE Trans Inf Forensics Secur 12(3):676–688

Zhang Y, Deng Robert H, Ximeng L, Dong Z (2018) Blockchain based efficient and robust fair payment for outsourcing services in cloud computing. Inf Sci 462:262–277

Zhang Y, Deng R, Zheng D, Li J, Pengfei W, Cao J (2019) Efficient and robust certificateless signature for data crowdsensing in cloud-assisted industrial iot. IEEE Trans Ind Inform 15(9):5099–5108

Zhang Y, Deng R, Liu X, Zheng D (2018) Outsourcing service fair payment based on blockchain and its applications in cloud computing. IEEE Trans Ser Comput

Zhao M, Ding Y, Wang Y, Wang H, Wang B, Liu L (2019) A privacy-preserving tpa-aided remote data integrity auditing scheme in clouds. In International Conference of Pioneering Computer Scientists, Engineers and Educators, pp 334–345. Springer, Berlin

Zhao Q, Zhao C, Cui S, Jing S, Chen Z (2020) Privatedl: Privacy-preserving collaborative deep learning against leakage from gradient sharing. Int J Intell Syst