Distributed additive encryption and quantization for privacy preserving federated deep learning

Zhu, Hangyu ; Wang, Rui; Jin, Yaochu; Liang, Kaitai; Ning, Jianting

DOI
10.1016/j.neucom.2021.08.062

Publication date
2021

Document Version
Final published version

Published in
Neurocomputing

Citation (APA)
Zhu, H., Wang, R., Jin, Y., Liang, K., & Ning, J. (2021). Distributed additive encryption and quantization for privacy preserving federated deep learning. *Neurocomputing*, *463*, 309-327. https://doi.org/10.1016/j.neucom.2021.08.062

Neurocomputing 463 (2021) 309–327

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Distributed additive encryption and quantization for privacy preserving federated deep learning

Hangyu Zhu [a,1], Rui Wang [b,1], Yaochu Jin [a,*], Kaitai Liang [b], Jianting Ning [c,d]

[a] Department of Computer Science, University of Surrey, Guildford, Surrey GU2 7XH, UK
[b] Department of Intelligent Systems, Delft University of Technology, Delft 2628XE, the Netherlands
[c] Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350007, PR China
[d] School of Information Systems, Singapore Management University, Singapore 178902, Singapore

ARTICLE INFO

Article history:
Received 21 May 2021
Revised 23 July 2021
Accepted 15 August 2021
Available online 18 August 2021
Communicated by Zidong Wang

Keywords:
Federated learning
Deep learning
Homomorphic encryption
Distributed key generation
Quantization

ABSTRACT

Homomorphic encryption is a very useful gradient protection technique used in privacy preserving federated learning. However, existing encrypted federated learning systems need a trusted third party to generate and distribute key pairs to connected participants, making them unsuited for federated learning and vulnerable to security risks. Moreover, encrypting all model parameters is computationally intensive, especially for large machine learning models such as deep neural networks. In order to mitigate these issues, we develop a practical, computationally efficient encryption based protocol for federated deep learning, where the key pairs are collaboratively generated without the help of a trusted third party. By quantization of the model parameters on the clients and an approximated aggregation on the server, the proposed method avoids encryption and decryption of the entire model. In addition, a threshold based secret sharing technique is designed so that no one can hold the global private key for decryption, while aggregated ciphertexts can be successfully decrypted by a threshold number of clients even if some clients are offline. Our experimental results confirm that the proposed method significantly reduces the communication costs and computational complexity compared to existing encrypted federated learning without compromising the performance and security.

1. Introduction

Federated learning (FL) [48] enables different clients to collaboratively train a global model by sending local model parameters or gradients to a sever, instead of the raw data. Compared to traditional centralized learning, FL cannot only address the problem of isolated data island, but also play an important role in privacy preservation. Consequently, FL has been deployed in an increasing number of applications in mobile platforms, healthcare, and industrial engineering, among many others [42,72].

However, FL consumes a considerable amount of communication resources. Model parameters or gradients need to be downloaded and uploaded frequently between the server and clients in each communication round, resulting in a longer operational stage especially for those large and complex models such as deep convolutional neural networks [35]. Much research work has been proposed to reduce the communication costs in the context of FL, including layer-wise asynchronous model update [12], reduction of the model complexity [79], optimal client sampling [56], and model quantization [4,69] based on the trained ternary compression [78]. It has been empirically and theoretically shown that using quantization in FL does not cause severe model performance degradation [4,14,17], and under certain conditions, quantization can even reduce weight divergence [69].

Although FL can preserve data privacy to a certain degree, recent studies [61,51,54,23,41] have shown that local data information can still be breached through the model gradients uploaded from each client. Under the assumption that the server is *honest-but-curious*, differential privacy (DP) [18] and homomorphic encryption (HE) [25] have been introduced into FL as additional privacy-preserving mechanisms. DP injects Gaussian [2] or Laplacian [61] noise into each uploaded model gradients [26,67], which is cost-effective and light-weighted. But the added noise has a negative impact on the model performance and cannot deal with data reconstruction attacks upon the model gradients [54].

Similar to FL, privacy preservation in distributed optimization [73,75] is also an intrinsic issue. Mao et al. [47] propose a privacy

* Corresponding author.
E-mail addresses: hangyu.zhu@surrey.ac.uk (H. Zhu), R.Wang-8@tudelft.nl (R. Wang), yaochu.jin@surrey.ac.uk (Y. Jin), Kaitai.Liang@tudelft.nl (K. Liang).
[1] Equally contributed.

https://doi.org/10.1016/j.neucom.2021.08.062
0925-2312/

preserving distributed optimization algorithm over time-varying directed communication networks by adding conditional noise to the exchanged states. Besides, Lu and Zhu [44] adopt HE to privately execute a distributed projected gradient-based algorithm on a set of agents. Different from the above work, Xu et al. [70] construct a federated data-driven evolutionary optimization framework to perform data-driven optimization without centrally storing the training data on the server.

HE is originally applied to outsourced data for privacy-preserving computation and supports direct ciphertext calculations. It can be roughly divided into partial homomorphic, somewhat homomorphic and fully homomorphic encryptions [25]. Among them, additive HE [52] can be seen as a partial HE that provides an efficient protection on gradients and enables gradient aggregation in FL, in which only the addition operation is involved. In an HE-based FL, each client encrypts its model gradients before uploading to the central server, where all encryptions can be directly aggregated. Phong et al. [54] adopted learning with error (LWE) based HE and Paillier encryption in distributed machine learning to safeguard communicated model parameters. But their method requires that all clients are honest, which is a very strong assumption and unrealistic for many real-world applications. Later, Truex et al. [63] performed threshold Paillier encryption [15] to construct a more practical system, which is more robust to malicious clients in the FL context. Besides, Ma et al. [45] proposed a privacy-preserving method for multi-party deep learning based on exponential ElGamal encryption and bilinear pairings. However, this work is based on an assumption that there is no collusion between server and clients.

Similar ideas that use HE in FL have also been presented in [46,28], which, however, incur a considerable increase in communication costs. More recently, Zhang et al. [74] proposed a batch-crypt scheme for batch gradients encoding and encryption without increasing the communication costs, but this method consumes huge amount of local computational resources.

Using HE and DP [63,77,32] together in FL has become a popular research topic nowadays, which can defend against attacks upon the shared global model on the server side. Nevertheless, DP protection in this framework has limited effect against inference attacks from the client side, since local data has been already "masked" by HE.

Besides HE and DP based protection methods, other techniques like secure masking has also been used. Bonawitz et al. [7] proposed a secure aggregation protocol by double masking uploaded local models with random numbers. In this scheme, keys are generated to do Diffle-Hellman key exchange without the help of a TTP. However, this method applies the Diffle-Hellman key exchange and Shamir secret sharing for every client, which requires a large amount of communication resources. Similarly, Kursawe et al. [36] proposed a single masking method for secure aggregation in smart grids without a TTP, assuming that grid collusion and party dropout do not exist, making it inapplicable to the FL environment.

Existing HE-based FL systems have the following two major drawbacks. First, most methods require a trusted third party (TTP) to generate and distribute key pairs, increasing topological complexity and attack surfaces of FL systems. In case the TTP is compromised, the secret messages will be revealed. Second, existing HE-based FL designs do not scale well to deep learning models containing a large number of parameters, because encryption and decryption of all trainable parameters are computationally prohibitive and uploading the entire encrypted model consumes a huge amount of communication resources. In addition, privacy protection techniques used in distributed optimization are usually unsuited for federated deep learning. The main reason is that the number of model parameters in distributed optimization is much less than that in federated deep learning. Therefore, there is no high demand for secure computation in distributed optimization. Take the method used in [44] as an example, each agent (client) generates its own key pairs to encrypt its state for n times without the help of TTP, where n is the number of agents. The payload of n times encryption is acceptable, since the state is just a scalar in distributed optimization. However, this becomes intractable in federated deep learning, as deep neural networks contain millions or even billions of model parameters.

To address the above challenges, this work aims to propose a practical and efficient privacy-preserving federated deep learning framework on the basis of additive ElGamal encryption [19] and ternary quantization of the local model parameters, DAEQ-FL for short. In DAEQ-FL, the global key pairs are collaboratively generated between the server and clients, and quantization makes it practical to encrypt deep neural networks in federated learning. The main contributions of the work are:

- We propose, for the first time, an efficient threshold encryption system with federated key generation and model quantization for federated deep learning systems. As a result, the number of model parameters to be encrypted and the communication costs for uploading the local models are considerably reduced, and an extra TTP is no longer required.
- An approximate model aggregation method is developed for the server to separately aggregate the uploaded ciphertexts and the quantized gradients, making it possible to download the aggregated ciphertexts to T (threshold value) qualified clients only for distributed partial decryption. Thus, the proposed approximate aggregation can further dramatically reduce the computational and communication cost for threshold decryption.
- Extensive empirical experiments are performed to compare the proposed method to threshold Paillier [63] with respect to the learning performance, communication cost, computation time and security. Our results confirm that even encoded with 10 bits, the proposed method can significantly decrease the computation time and communication cost with no or negligible performance degradation.

The remainder of the paper is structured as follows. Section 2 introduces the preliminaries of federated learning, deep neural network models, and the ElGamal encryption system. Details of the proposed methods, including federated key generation, encryption and decryption, ternary quantization, and model aggregation are provided in Section 3. Experimental results and discussions are given in Section 4. Finally, conclusion and future work are presented in Section 5.

2. Preliminaries

2.1. Federated learning

Unlike the centralized cloud model training that needs to collect raw data from different parties and store the data on a server, FL [72] is able to distributively learn a shared global model without accessing any private data of the clients. As shown in Fig. 1, at the t-th round of FL, K connected clients download the same global model θ_t from the server and update it by training with their own data. After that, trained local models or gradients will be uploaded back to the server for model aggregation. Therefore, the global model can be learned and updated while all the training data remain on edge devices.

FL aims to optimize a distributed loss function $\ell(\theta)$ as shown in Eq. (1),

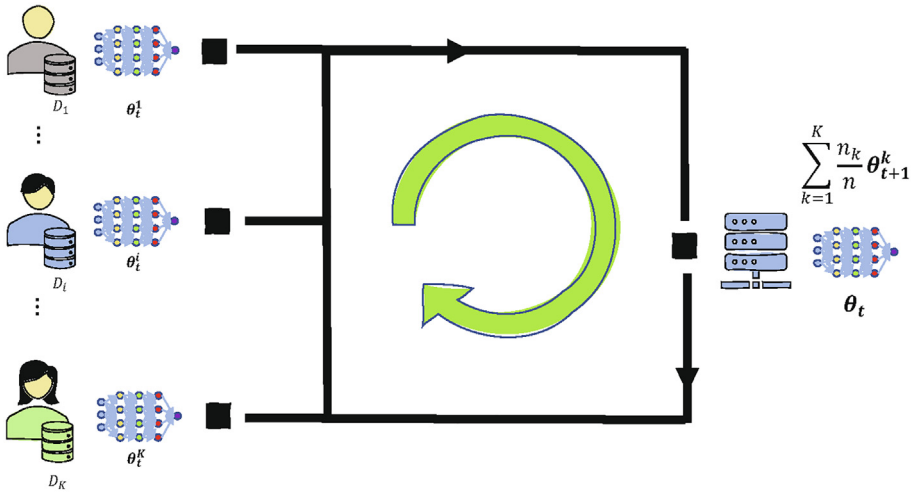


Fig. 1. Flowchart of federated learning. $\theta_t$are the global model parameters in the t-th communication round, n_k is the data size of client k, and K is the total number of clients. The global model parameters are randomly initialized at the beginning of the training and will be updated by aggregating the uploaded local models in each round.

$$\min_{\theta} \ell(\theta) = \sum_{k=1}^{K} \frac{n_k}{n} L_k(\theta), L_k(\theta) = \frac{1}{n_k} \sum_{i \in P_k} \ell_i(\theta, x_i) \tag{1}$$

where k is the index of K total clients, $L_k(\theta)$ is the loss function of the k-th local client, n_k equals to the local data size, and P_k is the set of data indexes whose size is n_k, i.e., $n_k = |P_k|$. Note that the training data x_i on each client k may not satisfy the independent and identically distributed assumption, i.e., non-IID.

Although the data on the clients do not need to be shared, FL is still subject to security risks, since the model gradients naturally contain information of the training data. It has been theoretically proved in [54] that only a portion of the gradients may lead to the leakage of private information of the local data. If we assume the cost function to be a quadratic function, the corresponding gradients can be calculated in Eq. (2).

$$\begin{aligned} &J(\theta, x) \stackrel{\text{def}}{=} (h_\theta(x) - y)^2 \\ &g_k = \tfrac{\partial J(\theta,x)}{\partial \theta} = 2(h_\theta(x) - y)\sigma'\left(\sum_{i=1}^{d} x_i w_i + b\right) \cdot x_k \end{aligned} \tag{2}$$

where g_k and x_k are the k-th feature of gradient $g \in R^d$ and the input data $x \in R^d$, respectively. Since the product $2(h_\theta(x) - y)\sigma'\left(\sum_{i=1}^{d} x_i w_i + b\right)$ is a real scalar number, the gradient is in fact proportional to the input data. As a result, uploading model gradients cannot completely prevent local data from being revealed and enhanced protection techniques are required for secure FL systems.

2.2. Deep learning

Deep learning has been deployed in the fields of computer vision, speech recognition and many other areas [27,37,57,16,50]. The word 'deep' means that neural network (NN) models, such as convolutional neural networks (CNNs) [38] and recurrent neural networks (RNNs) [58], used in deep learning always contain multiple hidden layers.

For a typical supervised learning [49], the training purpose is to minimize the expected distance between a desired signal y (e.g., a label in classification) and a predicted value $\hat{y}$, which is often represented by a so-called loss function $\ell(y, \hat{y})$ as shown in Eq. (3), where θ is the trainable model parameters we want to optimize.

$$\min_{\theta} \ell(\theta) = \frac{1}{N} \sum_{i} \ell(y, \hat{y}|\theta, x_i) \quad x_i \in \{x_1, x_2 \ldots, x_N\} \tag{3}$$

The stochastic gradient descent (SGD) algorithm is the most widely used optimization method that calculates the partial derivatives of the loss function (3) with respect to each model parameter in θ. The model parameters will be updated by subtracting scaled calculated gradients as shown in Eq. (4),

$$\begin{aligned} g_t &= \nabla_\theta \ell(\theta, x) \\ \theta_{t+1} &= \theta_t - \eta g_t \end{aligned} \tag{4}$$

where η is the learning rate and g_t is the expected gradient over data samples x at the t-th iteration. The model update based on SGD in (4) is repeatedly performed until the model parameters converge.

DNNs often contain a large number of layers and training very deep models on big datasets is extremely time-consuming. Therefore, DNNs will incur excessive communication and computation costs when they are adopted in FL.

2.3. Homomorphic encryption and secret sharing

HE is the most widely used data protection technology in secure machine learning as it supports algebraic operations including addition and multiplication on ciphertexts. An encryption method is called partially HE if it supports addition or multiplication operation, and fully HE if it supports an infinite number of addition and multiplication operations. Without loss of security and correctness, additive HE fulfills that multiple parties encrypt message $C_i = Enc_{pk}(m_i)$ and decrypt $\sum_{i=1}^{n} m_i = Dec_{sk}(\prod_{i=1}^{n} C_i)$ using public key and secret key, respectively.

Among those well-studied HE techniques, ElGamal[19] is a multiplicative mechanism while Paillier [52] provides additive operations, which are based on discrete logarithm and composite degree residuosity classes, respectively. The former needs 256-bit key length to achieve the 128 bit security level, whereas the latter costs 3072 bits [5], implying that ElGamal is a computationally more efficient encryption and decryption method [33]. However, additional Cramer transformation [13] needs to be applied to ElGamal encryption so as to extend it to support additive operations.

Conventional HE is not well suited for distributed learning systems such as FL systems, since the ciphertexts on the server can be easily inferred, as long as one client uploads its private key to the

server. In order to mitigate this issue, Adi Shamir [60] proposed Shamir Secret Sharing (SSS) which splits a secret into n different shares. Consequently, T-out-of-n shares are needed to recover the secret. Based on the SSS and Diffie–Hellman (DH) security definition (refer to Appendix A in the Supplementary material), Feldman proposed verifiable secret sharing (VSS) [20], which adds a verification process during sending shares.

But Feldman VSS only allows trusted clients to share secrets and it will fail to generate correct key pairs if there are adversaries in the system. To address this limit, Perdersen [53] proposed a novel VSS that is able to detect and exclude adversarial clients. On the basis of the above two VSS methods, Gennaro et al. [24] introduced a secure distributed key generation (DKG) for discrete logarithm based crytosystems.

We propose a federated key generation (FKG) based on DKG for model parameter encryption in FL and the details of FKG will be given in Section 3.

3. Our proposed system

The motivation of the proposed privacy preserving system is to remove the requirement of a TTP commonly used in FL for generating key pairs for encryption, and the whole system should be robust to malicious clients. A key component for the proposed system is federated key generation (FKG) that allows the server and clients collaboratively generate key pairs without a TTP. To this end, additive discrete logarithm based encryption is adopted to achieve secure model aggregation. In addition, a fixed point encoding method is implemented to encode the plaintext. In order to decrease computational and communication resources, ternary gradient quantization and approximate model aggregation are further introduced. In the following, we elaborate each of our main components and present a description of the proposed overall DAEQ-FL system. Finally, a brief discussion is given to compare our DAEQ-FL with existing encryption based FL systems.

3.1. Threat model

The potential threats considered in this work include those from the server, clients and outsiders in the FL proposed DAEQ-FL system. Specifically, in key generation period, we assume that at least T-out-of-n clients generate key pairs honestly. The remaining clients could be malicious and might try to steal keys from the honest parties or make the generated keys incorrect for encryption and decryption. In the training period, we consider the server and clients are honest-but-curious, which means they strictly follow the FL algorithm but try to infer extra information from received information. Our main goal is to prevent clients data from being leaked, but the situation in which malicious clients attempt to deteriorate global model performance [11,43] is not considered here. Besides, we assume that outsiders are passive attackers who could eavesdrop communication channels during the whole FL process.

3.2. Federated key generation

The proposed FKG is a variant of DKG [24], which is based on Pedersen VSS and Feldman VSS that can verify key shares in a secure way for successful key generations. The main steps of FKG are described in **Algorithm 1**.

Algorithm 1 Federated Key Generation. $\mathbb{G}$ is a cyclic group, p and q are large prime numbers, g is a generator, $y \in \mathbb{G}$, N is the number of total clients, C is the fraction of clients participating the current round, i is the client index, and T is the threshold value.

1: ***Server distributes public parameters*** $< p, q, g, y >$
2: **for** Each FL round $t = 1, 2, \ldots$ **do**
3: $n = C * N$
4: Select threshold value $T > n/2$
5: Client $i \in \{1, \cdots, n\}$ perform **Pedersen VSS**
6: Collect the number of complaints cpt_i for client i
7: **for** Client $i \in \{1, \cdots, n\}$ **do**
8: **if** $cpt_i > T$ **then**
9: Mark client i as disqualified
10: **else**
11: Client i uploads $f_i(j)$ and
12: **if** Eq. (6) is satisfied **then**
13: Mark client i as qualified (QUAL)
14: **else**
15: Mark client i as disqualified
16: **end if**
17: Mark client i as QUAL ▷$T \leqslant$—QUAL—$\leqslant n$
18: **end if**
19: **end for**
20: Client $i \in$ QUAL perform **Feldman VSS**
21: Collect complained client index in O,
22: **for** Each client $i \in O$ **do** ▷$|O| < T$
23: Set $counter = 0$
24: **for** Each client $j \in$ QUAL but $j \notin m$ **do**
25: Client j uploads $f_j(i)$ and $f'_j(i)$
26: **if** Eq. (7) is satisfied **then**
27: $counter = counter + 1$
28: **end if**
29: **if** $counter \geqslant T$ **then**
30: Break
31: **end if**
32: **end for**
33: Retrieve $f_i(z)$ and A_{i0} ▷$A_{i0} = g^{a_{i0}} (\mathrm{mod} p)$
34: **end for**
35: Generate global public key $h = \prod_{i \in QUAL} A_{i0} = g^x$
36: **end for**

Assume that the server in DAEQ-FL is *honest-but-curious*, and there are at least T-out-of-n ($T > n/2$) honest clients. Before key pair generation, the server needs to generate and distribute four public parameters p, q, g and y, where q is the prime order of cyclic group $\mathbb{G}$, p is a large prime number satisfying $p - 1 = rq$, r is a positive integer, g and y are two different random elements in $\mathbb{G}$.

For Pedersen VSS executed in line 5 of **Algorithm 1**, each participating client i in the t-th round generates two random polynomials $f_i(z)$ and $f'_i(z)$ over $\mathbb{Z}_q^*$ of order $T - 1$ as shown in Eq. (5).

$$\begin{aligned} f_i(z) &= a_{i0} + a_{i1} z + \ldots + a_{iT-1} z^{T-1} (\mathrm{mod} q) \\ f'_i(z) &= b_{i0} + b_{i1} z + \ldots + b_{iT-1} z^{T-1} (\mathrm{mod} q) \end{aligned} \tag{5}$$

Let $z_i = a_{i0} = f_i(0)$ be the locally stored private key. Client i broadcasts $C_{ik} = g^{a_{ik}} y^{b_{ik}} (\mathrm{mod} p)$ and sends shares $s_{ij} = f_i(j), s'_{ij} = f'_i(j)$ to client j $(j \in n)$, then client j verifies the received shares by *Pedersen commitment*:

$$g^{s_{ij}} y^{s'_{ij}} = \prod_{k=0}^{T-1} (C_{ik})^{j^k} (\mathrm{mod} p) \tag{6}$$

Due to the hiding and binding properties of Pedersen commitment [53], it is impossible for adversaries, if any, to guess the real a_{ik} and b_{ik} through $C_i k$ or to find another a pair of s_{ij} and s'_{ij} that can satisfy Eq. (6). In addition, based on our previous security assumption and the principle of SSS, it is infeasible to reconstruct the private keys of any honest clients even if the system contains $n - T$ malicious clients.

Each client sends a complaint of client i to the server if any shares s_{ij} and s'_{ij} received from client i do not satisfy Eq. (6). Once the server receives more than T complaints against client i (line 6 in **Algorithm 1**), this client will be immediately disqualified. Besides, as long as client i is complained by any client j, where $j \in \{1, \cdots, n\}$, the corresponding shares s_{ij} and s'_{ij} are required to upload to the central server for Pedersen commitment (Eq. (6)) verification. If any verification fails, client i would be marked as disqualified.

However, Pedersen VSS cannot guarantee correct global public key generation, since malicious clients can still corrupt the generation process by broadcasting fake A_{i0} (line 33 in **Algorithm 1**). Therefore, Feldman VSS is used in addition to Pedersen VSS to ensure that all the QUAL clients broadcast correct A_{i0} for in the proposed FKG.

Similarly, to implement Feldman VSS (line 20 in **Algorithm 1**), each client j ($j \in$ QUAL) broadcasts $A_{ik} = g^{a_{ik}} (\mathrm{mod} p)$ and verifies Eq. (7).

$$g^{s_{ij}} = \prod_{k=0}^{T-1} (A_{ik})^{j^k} (\mathrm{mod} p) \tag{7}$$

If shares of client i satisfy Eq. (6) but not Eq. (7), client j will send a complaint to server. Then the server requires T QUAL clients to upload their shares $f_i(j), j \in t$ to retrieve the random polynomial $f_i(z)$ of client i by Lagrange interpolation function [6] as show in Eq. (8).

$$\begin{aligned} &\lambda_j = \prod_{k \neq j} \tfrac{z-k}{j-k}, k \in T, j \in \mathrm{QUAL} \\ &f_i(z) = \sum_{j \in \mathrm{QUAL}} \lambda_j f_i(j) \end{aligned} \tag{8}$$

Finally, the server can generate the global public key through broadcasting A_{i0} from all QUAL clients in Eq. (9), where x is in fact the global private key. And then, the public key h will be shared to all QUAL clients.

$$\begin{aligned} &h_i = A_{i0} = g^{z_i} (\mathrm{mod} p) \\ &h = \prod_{i \in \mathrm{QUAL}} h_i = g^{\sum_{i \in \mathrm{QUAL}} z_i} = g^x (\mathrm{mod} p) \end{aligned} \tag{9}$$

Pedersen's VSS and Feldman's VSS for FKG require at most $768NT + 64N(N-1)$ bytes of communication, where N is the total number of clients and T is the threshold.

3.3. Additive Discrete Logarithm Based Encryption

To be fully compatible with FKG, which is adapted from DKG to the FL environment, additive discrete logarithm based encryption is employed based on ElGamal encryption [19].

The original ElGamal encryption works as follows:

- Parameters generation: Generate three parameters p, q and g, where q is the prime order of a cyclic group $\mathbb{G}$, p is a large prime number satisfying $q|p-1$, and g is a generator of $\mathbb{G}$.
- Key generation: Select a random number $x, x \in \mathbb{Z}_q^*$ as the secret key, and then compute $h = g^x (\mathrm{mod} p)$ to be the public key.
- Encryption: To encrypt a message $m \in \mathbb{Z}_p^*$, choose a random number $r \in \mathbb{Z}_q^*$ as a ephemeral key, calculate two ciphertexts as $< c_1 = g^r (\mathrm{mod} p), c_2 = m h^r (\mathrm{mod} p) >$.
- Decryption: The plaintext message m can only be decrypted if the private key x is available by computing Eq. (9),

$$\frac{c_2}{c_1^x} = \frac{m h^r}{(g^r)^x} = \frac{m g^{xr}}{g^{rx}} (\mathrm{mod} p) \equiv m \tag{10}$$

Therefore, the original ElGamal is a multiplicative HE satisfying: $\mathrm{Enc}(m1) * \mathrm{Enc}(m2) = \mathrm{Enc}(m1 * m2)$, because $m_1 h^{r_1} * m_2 h^{r_2} = m_1 m_2 h^{r_1 + r_2} (\mathrm{mod} p)$. Since model aggregation on the server in FL performs the addition operation, we can apply Cramer transformation [13] on ElGamal encryption by simply converting the plaintext m into $m' = g^m (\mathrm{mod} p)$. Consequently, the original ElGamal encryption becomes a discrete logarithm based additive HE, as shown in Eq. (11).

$$\begin{aligned} \mathrm{Enc}(m_1) * \mathrm{Enc}(m_2) &= g^{m_1} h^{r_1} * g^{m_2} h^{r_2} \\ &= g^{m_1 + m_2} h^{r_1 + r_2} (\mathrm{mod} p) \end{aligned} \tag{11}$$

A security analysis is described in Appendix B of the Supplementary materials.

3.4. Fixed point encoding method

Note that HE can be applied to integers only, however, model parameters or gradients are normally real numbers. Therefore, the real-values model parameters must be encoded before encryption.

The encoding method used in this work is straightforward, as shown in Eq. (12), where s_t is the maximum absolute of the gradients (will be introduced in Section 3.6), b is the encoding bit length, q is the above mentioned prime order of $\mathbb{G}$ and m is the encoded integer number.

$$\begin{aligned} &\widehat{m} = \mathrm{round}\left(s_t * 2^b\right), \widehat{m} \leqslant int_{\max} \\ &\mathrm{Encode}(s_t) = \widehat{m} (\mathrm{mod} q) = m \\ &\mathrm{Decode}(m) = \begin{cases} m * 2^{-b}, & m \leqslant int_{\max} \\ (m-q) * 2^{-b}, & m > q - int_{\max} \end{cases} \end{aligned} \tag{12}$$

$int_{\max}$ is the maximum positive encoding number defined by the server, which is one-third of q. If $m > q - int_{\max}$ ($\widehat{m}$ is negative), it should subtract q before multiplying 2^{-b}, because $\widehat{m} (\mathrm{mod} q) = q + \widehat{m}, \widehat{m} < 0$. Since the bit length of $int_{\max}$ is always set to be much larger than the encoding bit b, sufficient value space can be reserved for encoding number summations (additive HE).

3.5. Brute force and log recovery

Using Cramer transformation needs to recover the desired m from m' to solve the so-called discrete logarithm hard problem after decryption. Here, we propose two techniques, namely brute force and log recovery, to solve this problem.

Brute force recovery simply tries different m from 0 to $q-1$, and the correct m is found only if $m' = g^m (\mathrm{mod} p)$. Thus, a maximum of q trials are needed to solve DLHP in the worst case. Fortunately, the absolute values of all the model gradients in DNNs are

always less than 0.1, and therefore, they can be encoded with a b bit-length fixed point integer number. It takes about 2^b times to find the correct m if $\widehat{m}$ is positive. Note that the quantization method we employ can guarantee that $\widehat{m}$ is positive, which will be introduced later.

The log method consumes almost no additional recovery time by calculating $\log_g(g^m) = m$ directly. However, this only works when $g^m < p$. In order to ensure the best encoding precision (the encoded m should be as large as possible), we minimize g to $g_0 = 2$. And the encoded number m must be less than the bit length of a large prime number p, which means the encoding precision is restricted by the security level. Besides, the security level will not be reduced by selecting a small fixed g_0.

Both the brute force and log recovery are described in **Algorithm 2**

Algorithm 2 Plaintext Recovery. q is a prime order of the cyclic group $\mathbb{G}$, p is a large prime number satisfying $p-1|q$, g is a random element in $\mathbb{G}$, m is the message to be recovered, $int_{\max}$ is the maximum positive encoded number.

Brute Force Recovery:
$g_0 = g$
Given decrypted plaintext $m' = g_0^m (\mathrm{mod} p)$, $m \leqslant int_{\max}$
for j from 0 to $q-1$ **do**
  if $g_0^j (\mathrm{mod} p) == m'$ **then**
    $m = j$
    Break
  else
    Continue
  end if
end for
Return m

Log Recovery:
$g_0 = 2$
Given decrypted plaintext $m' = g_0^m (\mathrm{mod} p)$, $g_0^m \leqslant p$
$m = \log_{g_0}(g_0^m)$
Return m

3.6. Ternary gradient quantization

Encrypting and decrypting all elements of the model gradients have several shortcomings. First, performing encryption on local clients is computationally extremely expensive, causing a big barrier for real world applications, since usually the distributed edge devices do not have abundant computational resources. Second, uploading model gradients in terms of ciphertext incurs a large amount of communication costs. Finally, the first two issues will become computationally prohibitive when the model is large and complex, e.g., DNNs.

In order to tackle the above challenges, we introduce ternary gradient quantization (TernGrad) [68] to drastically reduce computational and communication costs for encryption of DNNs. TernGrad compresses the original model gradients into ternary precision gradients with values $\in \{-1, 0, 1\}$ as described in Eq. (13),

$$\begin{aligned} \tilde{g_t} &= s_t \cdot \mathrm{sign}(g_t) \cdot b_t \\ s_t &= \max(\mathrm{abs}(g_t)) \end{aligned} \tag{13}$$

where g_t is the full precision model gradients at the t-th iteration, $\tilde{g_t}$ is the quantized gradients, s_t (a scalar larger than 0) is the maximum absolute element among g_t, and the *sign* function transfers g_t into binary precision with values $\in \{-1, 1\}$. Finally, b_t is a binary tensor whose elements follow the Bernoulli distribution [64].

$$\begin{aligned} Pr(b_{tk} = 1|g_t) &= |g_{tk}|/s_t \\ Pr(b_{tk} = 0|g_t) &= 1 - |g_{tk}|/s_t \end{aligned} \tag{14}$$

where b_{tk} and g_{tk} is the k-th element of b_t and g_t, respectively, and the product of $\mathrm{sign}(g_t)$ and b_{tk} is a ternary tensor ($\{-1, 0, 1\}$) representing the model training direction. According to Eq. (4), the full precision model parameters in TernGrad is updated as shown in Eq. (15).

$$\theta_{t+1} = \theta_t - \eta(s_t \cdot \mathrm{sign}(g_t) \cdot b_t) \tag{15}$$

Since s_t is a random variable depending on input x_t and the model weights θ_t, Eq. (14) can be re-written into Eq. (16).

$$\begin{aligned} Pr(b_{tk} = 1|x_t, \theta_t) &= |g_{tk}|/s_t \\ Pr(b_{tk} = 0|x_t, \theta_t) &= 1 - |g_{tk}|/s_t \end{aligned} \tag{16}$$

The unbiasedness of the ternary gradients can be proved as shown in Eq. (17).

$$\begin{aligned} \mathbb{E}(s_t \cdot \mathrm{sign}(g_t) \cdot b_t) &= \mathbb{E}(s_t \cdot \mathrm{sign}(g_t) \cdot \mathbb{E}(b_t|x_t)) \\ &= \mathbb{E}(g_t) \end{aligned} \tag{17}$$

The TernGrad algorithm is adopted in the proposed DAEQ-FL to significantly reduce the computational cost for encryption on local devices and the communication costs for passing the encrypted model parameters between the clients and the server. Before performing encryption on the local clients, the model gradients are decomposed into two parts: one is the positive scalar s_t and the other is the ternary model gradients $\mathrm{sign}(g_t) \cdot b_t$. And only one scalar s_t in each layer of the model needs to be encrypted, thereby dramatically decreasing the computation costs and encryption time.

The ternary gradients do not need to be encrypted, because adversaries, if any, can only derive parts of gradients sign information from them. It should be noted that existing popular attacks [54,81,76,22,21,66,29] on FL do not have ability to retrieve the original data from ternary gradients. In addition, separately uploading encrypted s_t and ternary gradients reduce the total amount of uploads to 16 times smaller than the original. Another advantage is that it can make brute force recovery much faster, since the scalar s_t in Eq. (12) can never be negative.

3.7. Approximate model aggregation

For model aggregation on the server, the received encrypted s_t (t-th round in FL) should multiply its corresponding ternary gradients before weighted averaging (Eq. (18)).

$$\begin{aligned} g^i_{(t,\mathrm{tern})} &= \mathrm{sign}(g^i_t) \cdot b^i_t \\ \mathrm{Enc}\left(g^{\mathrm{global}}_t\right) &= \sum_i \frac{n_i}{n} \left(\mathrm{Enc}(s^i_t) * g^i_{(t,\mathrm{tern})}\right) \end{aligned} \tag{18}$$

Decryption of the aggregated global gradients $\mathrm{Enc}\left(g^{\mathrm{global}}_t\right)$ requires to traverse each single ciphertext, which is extremely time-consuming, making it less practical even if the server often possess abundant computation resources. Therefore, this work proposes an approximate aggregation method (Fig. 2) so as to reduce the decryption time. The basic idea is to separately aggregate the encrypted scalar and related ternary gradients, as shown in Eq. (19).

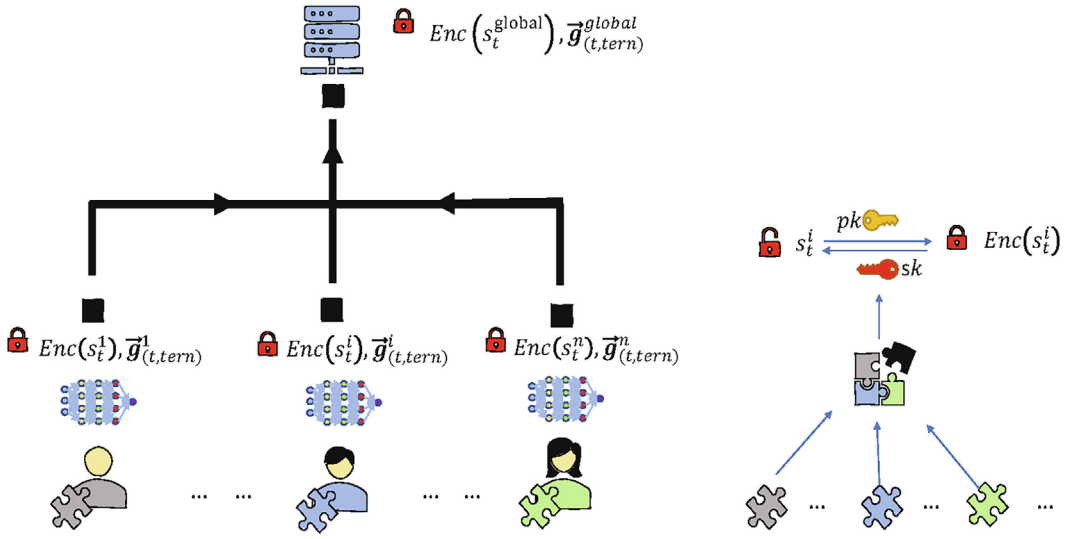


Fig. 2. Encryption with TernGrad and model aggregation approximation. The $\text{Enc}\left(s_t^{\text{global}}\right)$ is aggregated over the uploaded $\text{Enc}(s_t^i)$ from the participating clients.

$$\begin{aligned} \text{Enc}\left(s_t^{\text{global}}\right) &= \prod_i \text{Enc}\left(s_t^i * \tfrac{n_i}{n}\right) = \text{Enc}\left(\sum_i s_t^i * \tfrac{n_i}{n}\right) \\ g^{\text{global}}_{(t,\text{tern})} &= \sum_i g^i_{(t,\text{tern})} \end{aligned} \tag{19}$$

where i is the client index, n_i is the local data size and n is the global data size. And $\text{Enc}\left(g_t^{\text{global}}\right) = \text{Enc}\left(s_t^{\text{global}}\right) * g^{\text{global}}_{(t,\text{tern})}$, if $s_t^1 = s_t^2 = \ldots = s_t^n$. However, this condition is hard to satisfy and the bias $g_t^{\text{global}} - s_t^{\text{global}} * g^{\text{global}}_{(t,\text{tern})}$ is difficult to estimate due to the random property of SGD. In reality, each client's local scalar satisfies $s_t^1 \approx s_t^2 \approx \ldots \approx s_t^n$, and our experimental results (Section II.B in the Supplementary material) confirm that the values of s_t of different clients are very similar, making this approximation reasonable.

Note that a small implementation trick used here is to do weighted averaging upon s_t before encryption, which helps reduce the differences between s_t^i and avoid overflow of the encrypted gradients during model aggregations on the server.

To update the global model, only two ciphertexts $\text{Enc}\left(s_t^{\text{global}}\right)$ of each layer need to be decrypted, which will be multiplied by the global ternary gradients afterwards as shown in Eq. (19)

$$\begin{aligned} s_t^{\text{global}} &= \text{Dec}\left(\text{Enc}\left(s_t^{\text{global}}\right)\right) \\ \theta_{t+1}^{\text{global}} &= \theta_t^{\text{global}} - \eta s_t^{\text{global}} * g^{\text{global}}_{(t,\text{tern})} \end{aligned} \tag{20}$$

3.8. Overall framework: DAEQ-FL

The overall framework, distributed additive ElGamal encryption and quantization for privacy-preserving federated deep learning, DAEQ-FL for short, is depicted in **Algorithm 3**. Note that our algorithm generates key pairs at the beginning of each round, considering that in practice the keys are often frequently changed. This is, however, not a mandatory requirement.

Algorithm 3 DAEQ-FL. p, q, g, y are key parameters introduced in **Algorithm 1**, pk is the global public key, $Qual$ are qualified clients, N is the total number of clients, C is the fraction of connected clients, E is the number of local epochs, B is the local batch data, θ_t is the global model parameters at the t-th FL round, T is the threshold value, and η is the learning rate.

Server:
Generate and distribute p, q, g, y and global model parameters θ_0
for each FL round $t = 1, 2, \ldots$ **do**
  Select $n = C \times N$clients, $C \in (0, 1)$
  Select $T > n/2$
  Generate pk by FKG among n clients in **Algorithm 1**
  for each client $i \in Qual$ in parallel **do**
    Download θ_t
    Do local **Training**
    Upload $c^i_{(t,1)}$, $c^i_{(t,2)}$ and $\Delta\theta^i_{(t,tern)}$
  end for
  $c_{(t,1)} = \prod_i c^i_{(t,1)} (\text{mod} p)$
  $c_{(t,2)} = \prod_i c^i_{(t,2)} (\text{mod} p)$
  $\Delta\theta_{(t,tern)} = \sum_i \Delta\theta^i_{(t,tern)}$
  Randomly select T $Qual$ clients
  foreach client $j \in T$ in parallel **do**
    Download $c_{(t,1)}$ and $c_{(t,2)}$
    Do ***Partial Decryption***
  end for
  $g_0^{Tm_t} = \prod_{j \in T} pd_j = g_0^{Tm_t} g^{\left(x - \sum_j \lambda_j x_j\right) T} (\text{mod} p)$
  Recover Tm_t by **Algorithm 2**
  $\theta_{t+1} = \theta_t - \Delta\theta_{(t,\text{tern})} * Tm_t / T$
end for

Client i**:**
// **Training:**
$\theta_t^i = \theta_t$
for each iteration from 1 to E **do**
  for batch $b \in B$ **do**
    $\theta_t^i = \theta_t^i - \eta \nabla L_i\left(\theta_t^i, b\right)$
  end for
end for
$\Delta\theta_t^i = \theta_t^i - \theta_t$

(*continued on next page*)

34: Quantize $\Delta\theta_t^i$ into s_t^i and $\Delta\theta_{(t,\text{tern})}^i$ in Eq. (13) and (18)
35: Encode $m_t^i = \text{round}\left(s_t^i * D_k/D * 2^l\right)(\text{mod}q)$ in Eq. (12)
36: Encrypt $c_{(t,1)}^i = g^{r_i}(\text{mod}p)$, $c_{(t,2)}^i = g_0^{m_t^i} pk^{r_i}(\text{mod}p)$ ▷r_i is a random number, $r_i \in \mathbb{Z}_q^*$
37: **Return** $c_{(t,1)}^i$, $c_{(t,2)}^i$ and $\Delta\theta_{(t,tern)}^i$ to server
38: **// Partial Decryption:**
39: $x_i = \sum_j s_{ji} = f_j(i)$, $i,j \in$ QUAL ▷$f_j(i)$ in **Algorithm 1**
40: Partial decrypt $pd_i = c_{(t,2)}/c_{(t,1)}^{\lambda_i x_i T}(\text{mod}p)$
41: **Return** pd_i

Note that the server can determine the threshold value T based on the number of participating clients n in each FL round ($T > n/2$). If the number of QUAL clients are less than T, the disqualified clients will be kicked out of the system and then the process is aborted and FKG is restarted. After FKG, each QUAL client i downloads the global model parameters θ_t and the public key pk for local training. Then the model gradients, obtained by subtracting the received global model θ_t from the local updated model θ_t^i, are converted into a real-valued coefficient s_t^i and a ternary matrix $\Delta\theta_{(t,\text{tern})}^i$ before performing ElGamal encryption. Two ciphertexts $c_{(t,1)}^i$ and $c_{(t,2)}^i$ together with a ternary gradient $\Delta\theta_{(t,\text{tern})}^i$ are then uploaded to the server for model aggregation as described in Section 3.7.

For decryption (Fig. 3), only two aggregated ciphertexts need to be downloaded to T QUAL clients for partial decryption and T partial decrypted ciphertexts pd_i are uploaded back to the server (line 16–20 in **Algorithm 3**). The server can easily get the plaintext $g_0^{Tm_t}$ by multiplying all received ciphertexts pd_i. s_{ji} is a share $f_j(i)$ (introduced in **Algorithm 1**) from client j to client i, $\lambda_i = \prod_{j\neq i} \frac{j}{j-i}$ is the Lagrange coefficient and $x = \sum_{j\in\text{QUAL}} z_j$ is the global private key. According to the property of SSS, at least T (threshold value) different $f_i(j), j \in T$ shares are needed to retrieve the local private key z_i. The reason why $x - \sum_i \lambda_i x_i = 0$ is proved below:

$$\begin{aligned}\sum_{i\in T}\lambda_i x_i &= \sum_{i\in T}\lambda_i \sum_{j\in\text{QUAL}} f_j(i) \\ &= \sum_{j\in\text{QUAL}}\sum_{i\in T}\lambda_i f_j(i) \\ &= \sum_{j\in\text{QUAL}} z_j = x\end{aligned} \tag{21}$$

One of the advantages of DAEQ-FL is that no parties within the FL system, including the server, can know the global private key x, which significantly enhances system security level. In addition, the extra communication resources are negligible, and only three ciphertexts $c_{(t,1)}, c_{(t,2)}$ and pd_i are transmitted between the server and each client i with the help of the TernGrad algorithm. Finally, DAEQ-FL is robust to possible disconnection of individual clients, since the ciphertext can be successfully decrypted so long as a minimum of T QUAL clients upload their pd_i.

3.9. Discussion

We list the general differences between our proposed system and four popular existing approaches in Table 1. From the table we can see that our DAEQ-FL has remarkable superiority, being a threshold based encryption system without an extra TTP and tolerating the existence of malicious clients. We here note that Truex et al. [63] also proposed a threshold based Paillier encryption system in FL, but it still requires a TTP for key generation. Some quantization technique is introduced in [74] for efficient encoding and encryption, but it is totally different from our ternary quantization methods.

Compared to LWE [55,9] causing excessive message expansion and functional encryption [8,40] with large computaional complexity, Paillier is a relatively lightweighted encryption scheme. Therefore, we compare Paillier with our system in the next section. It is worth considering that in the real world scenario, outsiders could tamper the messages communicating between server and clients, a signature scheme could be used here [19] based on keys generated by FKG. In addition, the double masking method proposed by Bonawitz et al. [7] needs to recover masking values as

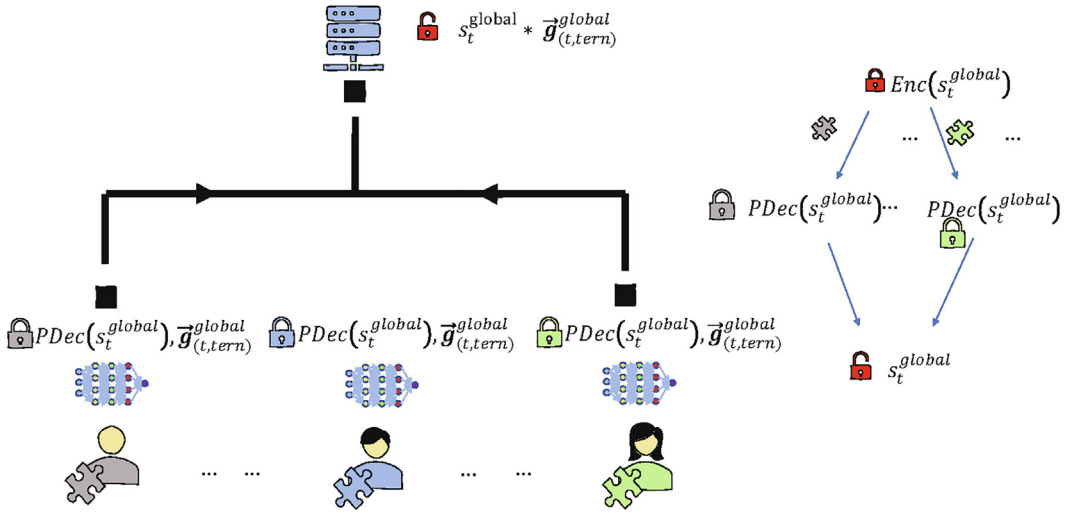


Fig. 3. The encrypted s_t^{global} is downloaded to T qualified clients for partial decryption, and then the partial decrypted ciphertexts are uploaded back to the server to retrieve the final plaintext.

Table 1
Comparison of encryption based privacy preserving FL systems.

Proposed systems	Threat model		Encryption scheme	Without TTP	Threshold Based
	Server	Client			
Phong et al. [54]	honest but curious	honest	Paillier, LWE	✗	✗
Truex et al. [63]	honest but curious	majority honest	Paillier	✗	✓
Xu et al. [71]	honest but curious	majority honest	functional encryption	✗	✗
Batchcrypt[74]	honest but curious	honest	Paillier	✓	✗
Ma et al. [45]	honest but curious	honest	ElGamal	✓	✗
DAEQ-FL(our system)	honest but curious	majority honest	ElGamal	✓	✓

long as one client is offline which is not computationally efficient. Our threshold based encryption scheme is more efficient and robust to this drop out problem, since it only require T-out-of-n clients to be online during decryption period.

4. Experimental results

In this section, we first introduce the datasets and models used in the experiments, together with all settings of the FL and encryption. We also present the communication and computation cost resulting from encryption, as well as the time consumption of the brute force recovery, followed by the analysis of approximate aggregation and a description of results on the model performances. Intel(R) Xeon(R) Gold 5218 CPU @ 2.30 GHz and NVIDIA Quadro RTX 6000 GPU are used in our experiments.

4.1. Dataset and model information

In our simulations, we use CNN for MNIST [39] digit number classifications, ResNet for CIFAR10 [34] image classifications and stacked LSTM [30] for Shakespeare [59] next word prediction task. All three datasets are non-iid among different clients.

MNIST is a 28x28 grey scale digit number image dataset containing 60,000 training images and 10,000 testing images with 10 different kinds of label classes (0∼9). All the clients' training data are distributed according to their label classes and most clients contain only two kinds of digits for non-iid partition.

CIFAR10 contains 10 different kinds of 50,000 training and 10,000 testing 32x32x3 images. Similar to MNIST, the whole training data are horizontally sampled and each client owns five different kinds of object images.

The Shakespeare dataset is built from the whole work of Wailliams Shakespeare. It has in total 4,226,073 samples with 1129 role players and the data samples of each role player represent the dataset on each client. Additionally, 90% of the user's data are randomly divided as the training data and the rest are testing data. This dataset is naturally non-iid and unbalanced, with some clients having few lines and others a large number of lines. In order to reduce the training time, we follow the method used in [10] to randomly select 5% of the total users and remove those containing less than 64 samples.

Note that we do not apply any data augmentation techniques [62,65] to boost the final global model performances to reduce local computational complexity, since the main purpose of this work is not to achieve the state of the art model performances in FL; instead, we aim to present a distributed encryption method for better privacy preservation in FL without considerably increasing the computational and communication costs.

A CNN model is adopted to train on MNIST in the FL framework, which contains two 3×3 convolution layers with 32 and 64 filter channels, respectively, followed by a 2×2 max pooling layer. And then, a hidden layer with 128 neurons is fully connected to the flattened output of the max pooling layer. Thus, the whole CNN model has 1,625,866 learnable parameters.

CIFAR10 dataset is to be learned by a ResNet model. The input images firstly pass through a 3x3 convolutional layer with 64 channels, followed by a batch normalization layer [31] with the Relu [3] activation function. Its output is connected to four sequentially connected block layers with 64, 128, 256, 512 filter channels, respectively. Each block layer contains two residual blocks containing two convolutional layers, each followed by a batch normalization layer and a shortcut connection. All the trainable parameters of the batch normalization layers are disabled, because they are observed to perform poorly with small batch sizes and non-iid data [31,80]. The full ResNet model has 11,164,362 trainable parameters.

The Shakespeare dataset is trained by a stacked LSTM model which contains two LSTM layers, each with 256 neurons. Since we use the module cudnnLSTM of Tensorflow [1], the layer bias is twice as large as the original LSTM layer. Thus, the full model contains 819,920 parameters.

4.2. Federated learning and encryption settings

In the experiments for image classification using CNN and ResNet, the FL system consists of total 20 clients, each containing 3000 and 2500 data samples for MNIST and CIFAR10, respectively. The total number of communication rounds is set to be 200 and all the clients are connected to the server in each communication round. In the experiments for language modeling using the LSTM, we randomly sample 5% of the entire role players (36 role users containing at least 64 samples) in Shakespeare dataset. Therefore, only 10 out of the 36 clients are randomly chosen to participate the training, following the settings in [10]. The total number of communication rounds is set to be 100.

We use SGD algorithm for all model training. For the CNN models, the number of local epochs is set to 2, the batch size is 50, and the learning rate is 0.1 with a decay rate of 0.995 over the FL rounds. We do not use any momentum for training the CNNs, while the momentum is set to be 0.5 for the ResNet. For the LSTM, the local epoch is set to 1, the batch size is 10, and the learning rate is 0.5 with a decay rate of 0.995.

The threshold rate is set to be 0.6 and the corresponding threshold value T is $0.6n$, where n is the number of connected clients in each communication round. Note that threshold T can be set to any value so long as it satisfies the condition $T > n/2$ so that the assumption that the number of benign clients is larger than that of malicious clients holds. The key size and group size of the distributed additive ElGamal encryption are set to 256 and 3072, respectively, to offer a 128-bit security level. Besides, the bit length b for encoding is chosen to be from 2 to 15. Due to the limited computation resources, a very large encoding bit length ($b > 15$) is not used here, since it will consume much more time for brute force recovery. Besides, for the log recovery, g_0 is set to 2 for fast encryption and the condition $g_0^m \leqslant p$ must be satisfied as described in

Table 2
Communication costs of one connected client for both encryption and partial decryption with 128-bit security level, # of ciphertexts means the number of transmitted ciphertexts for encryption and decryption, respectively.

Models	Enc Uploads (MB)	Dec Downloads (MB)	Dec Uploads (MB)	# of Ciphertexts
CNN (DAEQ-FL)	0.3876 + 0.0059	0.0059	0.0029	16 + 24
ResNet (DAEQ-FL)	2.6618 + 0.0161	0.0161	0.0081	44 + 66
LSTM (DAEQ-FL)	0.1955 + 0.0066	0.0066	0.0033	18 + 27
CNN (Paillier)	595.4099	595.4099	595.4099	1625866 + 3251732
ResNet (Paillier)	4088.5115	4088.5115	4088.5115	11164362 + 22328724
LSTM (Paillier)	300.2637	300.2637	300.2637	819920 + 1639840

Table 3
Runtime for encryption and decryption of one number using ElGamal and Paillier.

Algorithm	Enc Time (s)	Dec Time (s)
ElGamal	0.0029	0.0015
Paillier	0.0501	0.0141

Algorithm 2. Since the group size of p is 3072, the encoded number m must satisfy the condition $2^m \leqslant 3072$ and the corresponding encoding bit length satisfies the condition $b \leqslant 11.6$. Also considering the overflow problem previously discussed, the largest encoding bit length for log recovery is set to 10. As a result, the log recovery is used when the encoding bit length ranges from 2 to 10, while the brute force recovery method is adopted when the bit length is larger than 10.

4.3. Encryption cost and brute force recovery time

At first, we compare the communication costs between the proposed DAEQ-FL and a threshold based Paillier method in terms of encryption and recovery costs.

We experiment with three different models (CNN, ResNet and LSTM) in our DAEQ-FL system and compare them with the Paillier-based variants. As shown in Table 2, the communication cost of our system is dramatically less than those Paillier-based systems. The best case has been achieved in the LSTM, where each client consumes only 0.212 MB of communication costs in one round, whereas the Paillier system takes 900.7911 MB, which is about 4249x of our system. For training the CNN and ResNet, the proposed DEAQ-FL costs 0.4023 MB and 2.7021 MB, respectively, which accounts for approximately 0.023% and 0.022% of the Paillier based variants.

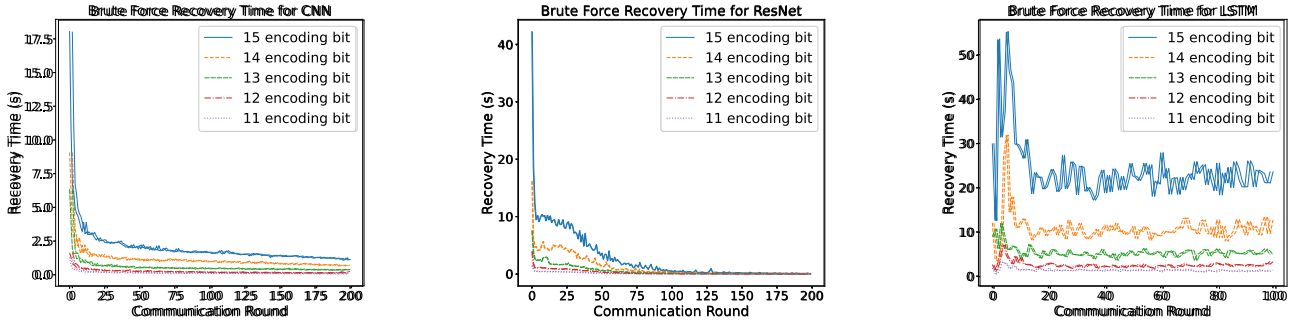
Next, we compare the runtime of the ElGamal encryption used in our system and the conventional Paillier method under the same security level. The runtimes of the two encryption methods for encrypting and decrypting one number are listed in Table 3, where both the key size of Paillier and the group size of ElGamal are 3072.

From the results in Table 3, we can observe that ElGamal is approximately 17 times and 10 times faster than Paillier for encryption and decryption, respectively. However, since the Cramer transformation needs extra brute force recovery time, in the following, we explore the brute force recovery time corresponding to different encoding bit lengths for CNN, ResNet18 and stacked LSTM, respectively. The comparative results are plotted in Fig. 4.

Because the larger the encoding bit length is, the more time the brute force recovery will consume. Here, we experiment with the encoding bit length starting from 15 bits to 2 bits. The results clearly show that the difference in computation times goes bigger with rising of encoding bit. Specifically, CNN, ResNet and LSTM spend at most 18.0467s, 42.2203s and 55.2088s on recovery, respectively.

The CNN and ResNet show similar recovery time profile over the communication rounds. Their brute force recovery time are very large in the beginning, and quickly drop over the communication rounds. This is attributed to the fact that the gradients of the model parameters of the SGD decrease quickly as the global model parameters converge. It is surprisingly to see that the recovery time for the ResNet becomes almost zero, which is smaller than that of the CNN after approximately 100 communication rounds. This means the model gradients of the ResNet become very small at the end of federated model training.

By contrast, the recovery time of the LSTM does not drop to zero and keeps fluctuating at a relatively high level, especially for the 15-bit length encoding. There are two reasons for the above observations. First, training of the LSTM involves large gradients of



(a) CNN for MNIST dataset (b) ResNet for CIFAR10 dataset (c) LSTM for Shakespeare dataset

Fig. 4. Brute force recovery time with different encoding bit lengths for different learning models.

Table 4
Brute force recovery time for 15 encoding bit length.

Models	Max (s)	Min (s)	Avg (s)
CNN	18.0467	1.0717	1.9786
ResNet	42.2203	0.0474	2.6491
LSTM	55.2088	12.6166	23.8876

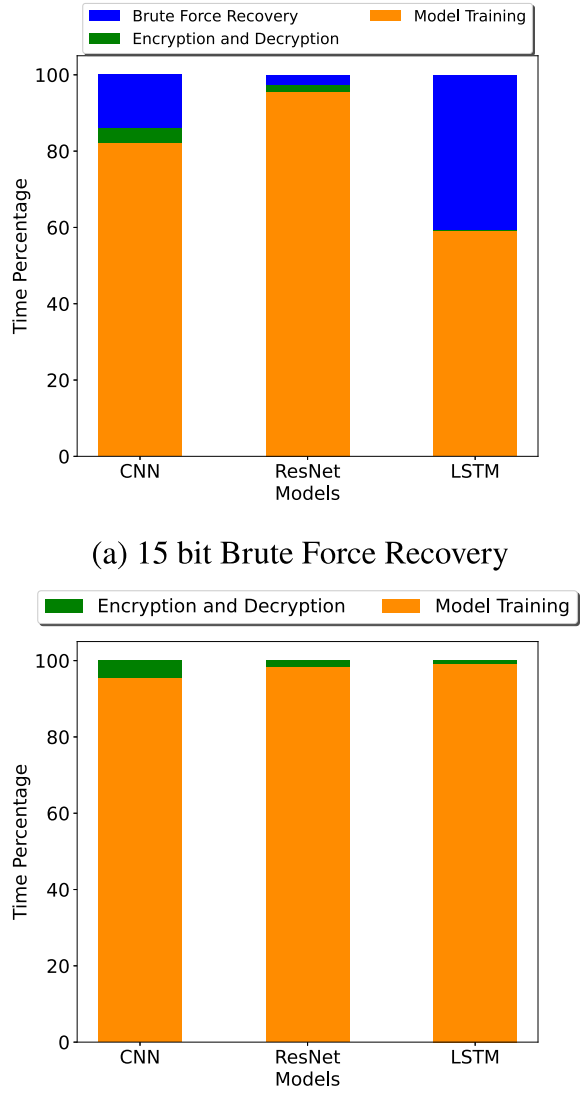


Fig. 5. Ratio of consumed time for model training, encryption and decryption, and brute force recovery for the 15-bit encoding length (a), and for the log recovery for the 10-bit encoding length (b) in one communication round.

recurrent connections, and those values are determined by the length of sequence. Second, the setting of the FL environment for the LSTM is very different from that of the CNN and ResNet, where only ten clients randomly participate global model aggregation in one communication round.

Table 4 presents the time consumption of the brute force recovery for 15-bit encoding length. From Fig. 5(a) we can find that the average brute force recovery time accounts for a great proportion of the total elapsed time in each communication round, especially for the LSTM.

Therefore, we can use the log recovery instead of the brute force recovery when the encoding bit length is smaller than or equal to 10 so that the recovery time and the encryption time become negligible (Fig. 5(b)). Since the group size of p is 3072 bit and g_0 is 2, the log recovery is not recommended when the plaintext message is larger than 3072 bit (11-bit encoding length). In order to avoid overflow, the maximum encoding bit length is set to be 10 in our simulations and the global performance drop caused by a low encoding bit length will be discussed in the next section.

4.4. Analysis of approximate model aggregation

For approximate aggregation analysis, we vary the degrees of data skew among each client, since the local gradient differences mainly come from different local data distributions. More specifically, there are three kinds of different local data distributions: 1) independent and identical distributed (iid): the training data is randomly allocated to each client, 2) non-iid with 5 classes: each client owns five different kinds of object images, 3) non-iid with 2 classes: each client owns two different kinds of images.

ResNet is adopted for this analysis, since it is the most complex model containing the largest trainable parameters in our experiments and the results derived from ResNet are more representative. In addition, we just display the maximum absolute elements s_t of the first convolutional layer and the last fully connected layer, because it is redundant and unnecessary to show total 22 s_t values of all 20 layers in one client. Except that, these experiments are performed under the federated encryption environment with our DAEQ-FL algorithm and two kinds of encoding bit length (10 and 15) are used here.

The purpose of this ablation study is to observe whether the values of s_t of different clients are similar. And if $s_t^1 \approx s_t^2 \approx \ldots s_t^n$ holds, the bias between g_t^{global} and $s_t^{\text{global}} * g_{(t,\text{tern})}^{\text{global}}$ in Eq. (19) of the original paper would be very small.

All experimental results with three different kinds of data distributions are shown in Fig. 6–8, respectively. It is clear to see that, s_t values from different clients do not show big differences with each other over communication rounds.

More specifically, for iid cases, the curves of different s_t values are nearly located at a single line. In other words, different s_t values have almost the same value and our proposed approximate aggregation algorithm has extremely small performance biases for iid data compared to original weighted averaging method.

For non iid cases with 5 kinds of local images, the curves show more variations than those of iid cases. However, even for the scenarios of 10 bit encoding length, the observed maximum distance between two s_t values of the first convolutional layer is less than 0.003 in Fig. 7(a), and the approximation biases caused by these small differences can be negligible.

The results of very extreme non iid cases with only 2 kinds of local images are also presented in Fig. 8. This case has an obvious difference that the s_t values of the last fully connected layer show more variations than those of two previous situations. And it is also clear to see that the maximum distance between two s_t values is around 0.005 in Fig. 8(d), thus, even for this extreme case, our proposed approximate aggregation algorithm would not bring in large performance biases and is still valid in DEAQ-FL algorithm.

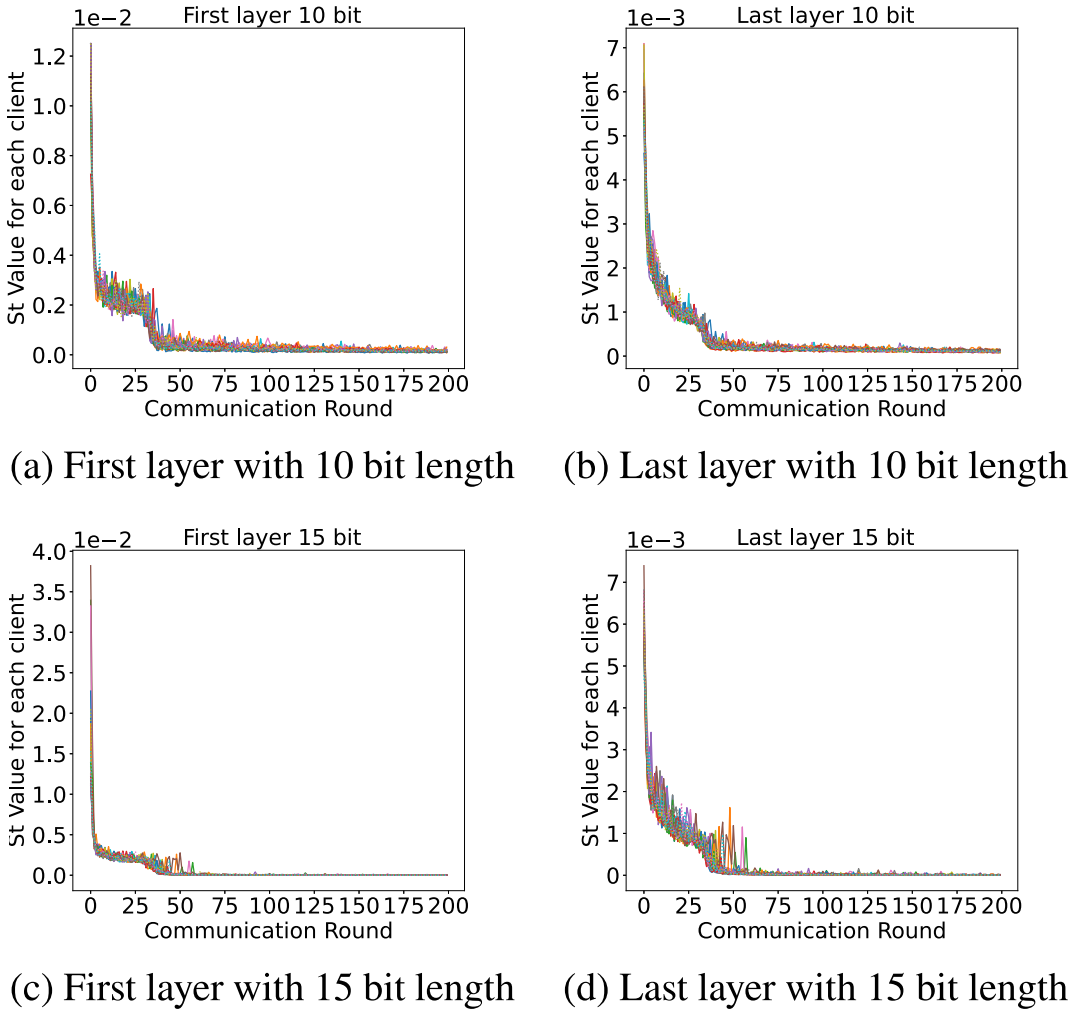


Fig. 6. s_t values of all connected clients with iid data.

4.5. Learning performance

In this section, we empirically examine influence of the TernGrad quantization, approximate aggregation and encoding length on the learning performance of the proposed DAEQ-FL system. Fig. 9 shows the test accuracy of the three models with or without encryption operations. For non encryption cases, 'Original' represents standard FL, 'TernGrad' means only quantization is used and 'TernGrad + Approx' uses both quantization and approximated aggregation technique. And for encryption cases, brute force recovery is used for 15 encoding bit length cases and log recovery is adopted for 10 bit length scenarios.

From these results, we can see that the test accuracy of the models of the original FL and four variants of the DEAG-FL have achieved almost the same performance (in particular the CNN, with 98.97% test accuracy). These results indicate that both the quantization and approximated aggregation have negligible impact on the test performance of the global model. Besides, neither the quantization or encryption has considerably slowed down the convergence speed over the communication rounds. It also can be seen that, the CNN and ResNet converge around the 25th round, while the LSTM is convergent around the 50th round.

More specifically, when the CNN trained on the MNIST dataset, the model performance does not degrade when the encoding bit length is decreased to 10. However, For the ResNet trained on CIFAR10, the test accuracy of the global model is reduce to 74.96% by using 10-bit encoding, which is approximately 1% lower than models without using encryption. The test accuracy of the LSTM models fluctuates between around 48% and 52%, regardless whether quantization or encryption are used or not. This can be due to the same reasons as discussed above.

Now we take a closer look at the learning performance of the global model of the proposed DAEQ-FL when the encoding bit length further decreases. The results for all models are shown in

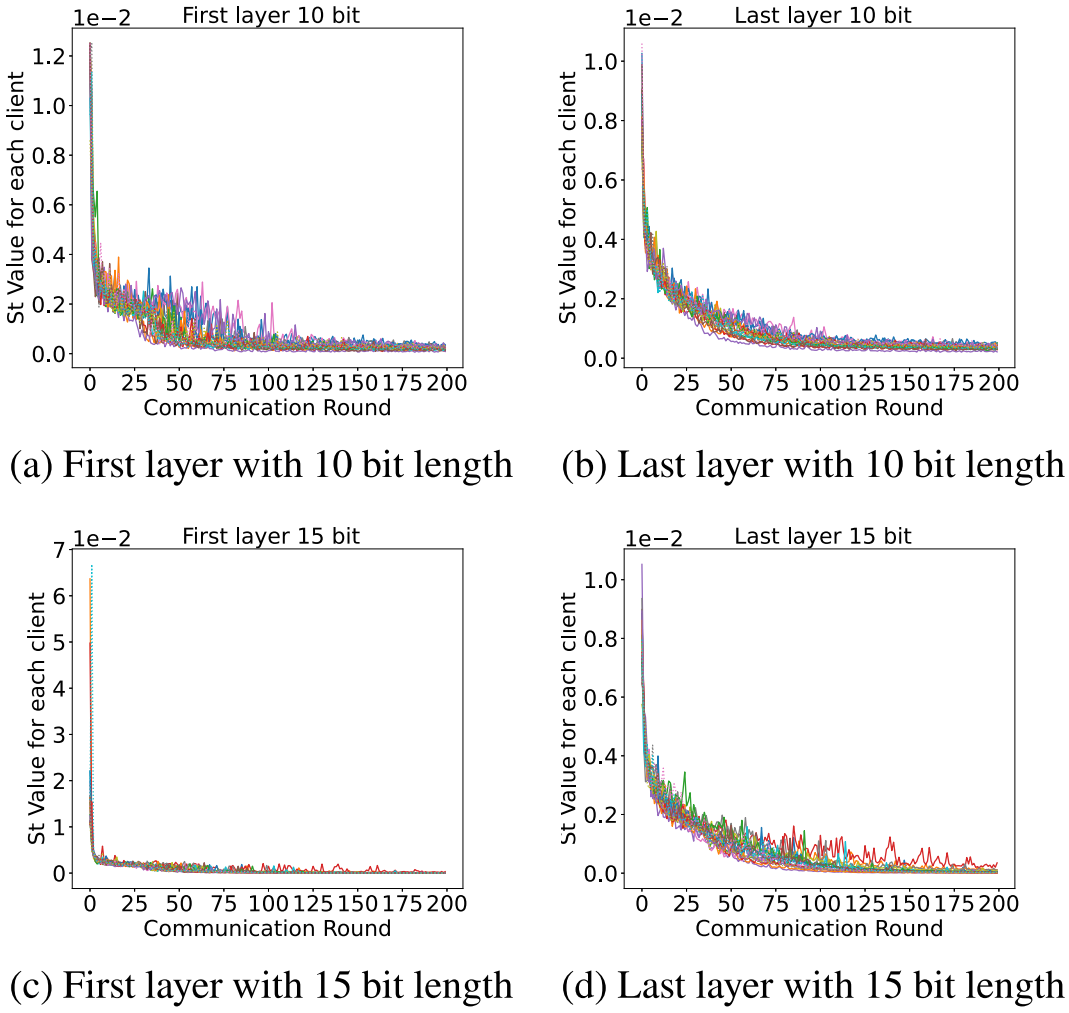


Fig. 7. s_t values of all connected clients with non iid data that each client contains only 5 different kinds of object images.

Fig. 10. It can be noticed that the convergence speed of the CNN and ResNet starts to decrease when the encoding bit length is smaller than 9. Besides, the model performance reduces dramatically when the encoding length is reduced to 7 bits or lower for CNN and to 8 bits or lower for the ResNet, respectively. Reduction of the encoding bit length does not cause clear model degradation for the LSTM until when only 2 bits are used for encoding and the accuracy is reduced to 43.15%. Nevertheless, we can still observe a slight drop in the convergence speed in the beginning of the communication rounds. The possible reason for this is that the model gradients of LSTM are large.

To take a closer look at the relationship between the model performance and computation time (related to the encoding length if the bit size is larger than 10), the test performances over different brute force recovery time are listed in Table 5. It is clear to see that the model test accuracy on three datasets does not have big variations with the increase of the brute force recovery time. On the CIFAR10 and MNIST datasets, however, the model performances have a significant drop from 7 bit encoding length. In this case, the log recovery method is adopted to alleviate the brute force recovery time. And all runtimes should be the same without considering the error in calculating the system computation time and other interference noise.

Overall, both TernGrad quantization and the approximate global model aggregation have little impact on the model performance, so long as the encoding bit length is not less than eight. Therefore, the proposed DEAQ-FL using the log recovery and an encoding length of 9 or 10 can achieve 128 bit security level with negligible degradation in performance and little increase in computational and communication costs for the CNN, ResNet and LSTM on the corresponding datasets studied in this work.

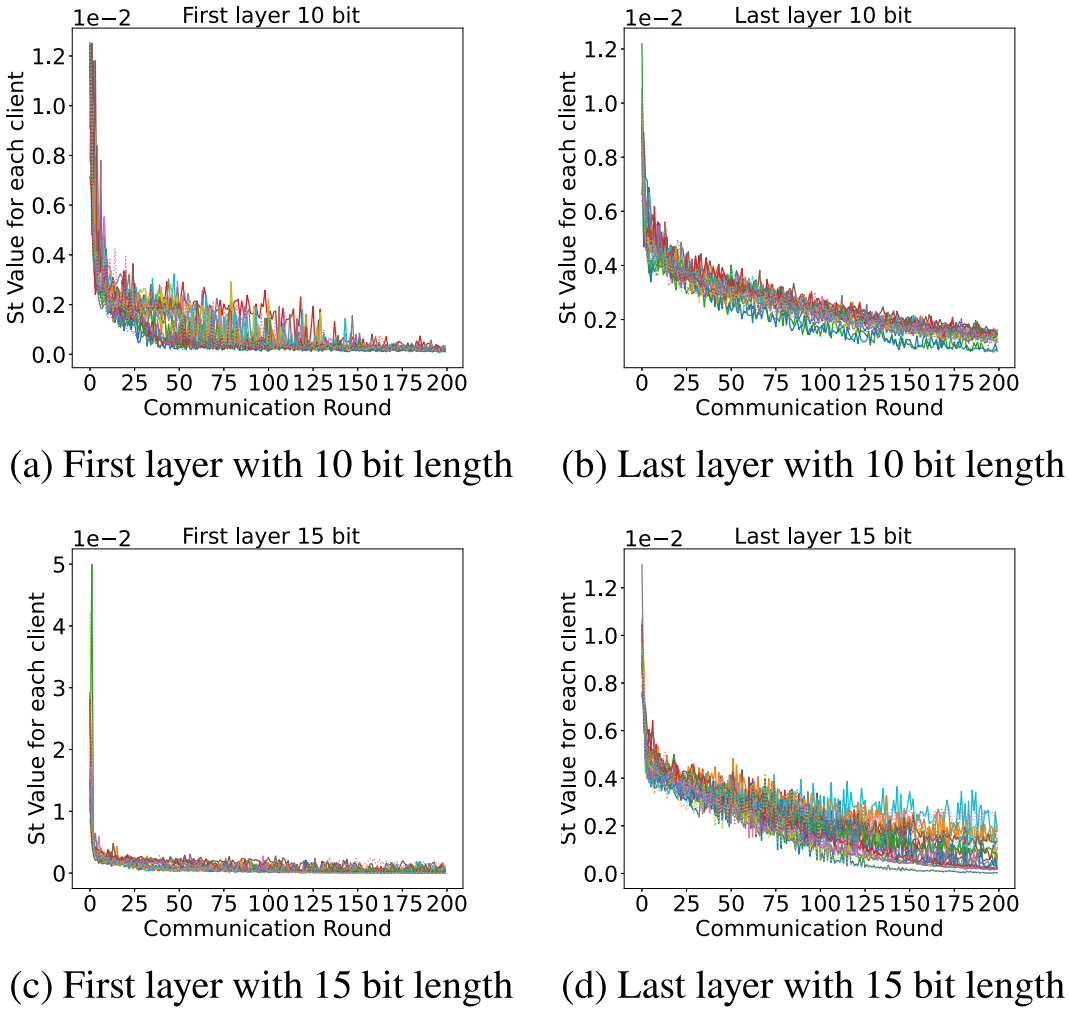


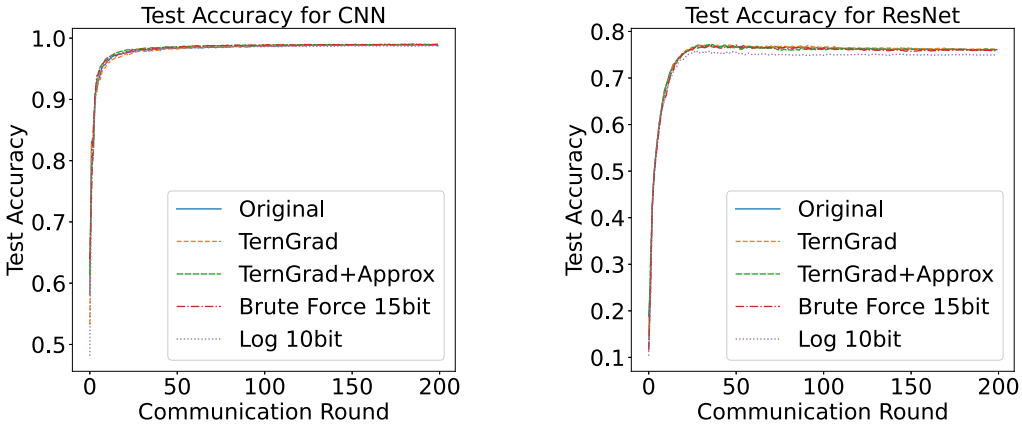
Fig. 8. s_t values of all connected clients with non iid data that each client contains only 2 different kinds of object images.

5. Conclusion and future work

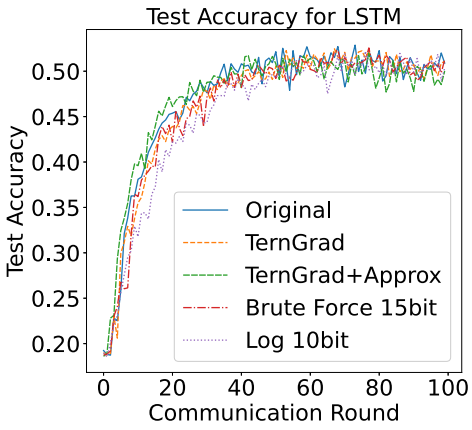
In this paper, we propose a privacy-preserving solution that makes use of distributed key generation and additive ElGamal encryption to protect gradients in the federated learning framework. To reduce computational and communication costs, we also introduce ternary quantization of the local models and approximate aggregation of the global model, making our solution practical in complex machine learning models, such as deep neural networks, in the context of gradient encryption. The proposed DAEQ-FL system does not rely on a TTP for key pair generations, which enables the system to tolerate a certain number of malicious clients.

DEAQ-FL can adopt the computationally efficient log recovery when the encoding bit length is less than eleven, and there is no noticeable learning performance degradation when ten bits are used for encoding (note only about 1% accuracy loss for ResNet compared to the results without doing any encryption on the Shakespeare dataset). However, the model learning performance starts to clearly deteriorate when the encoding length is less than eight. Thus, brute force recovery must be adopted when the encoding length is larger than eleven. Although a large encoding length can enhance the coding precision, the amount of recovery time will considerably increase. According to our experimental results, the global models trained with a maximum of fifteen bits can perform comparably to the non-encrypted FL, while the recovery time is still acceptable. Since the model gradients tend to decrease to zero during training, the brute force recovery time is expected to become much smaller as the global model converges.

Although the proposed method shows highly promising performance for encrypted federated deep learning, it may need to balance a trade-off between model performance and computation time needed for plaintext recovery when a small model (e.g., a logistic regression) is adopted. This is mainly because a smaller model will require a much longer encoding length (e.g., more than fifteen), making the brute force based recovery intractable. Therefore, our future work will be dedicated to the development of a distributed additive homomorphic encryption without recovery that can be used in federated learning systems.



(a) CNN for MNIST dataset (b) ResNet for CIFAR10 dataset



(c) LSTM for Shakespeare dataset

Fig. 9. The test accuracy of the global model for CNN, ResNet and LSTM in five different settings.

CRediT authorship contribution statement

Hangyu Zhu: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Rui Wang:** Conceptualization, Methodology, Formal analysis, Investigation, Writing - original draft. **Yaochu Jin:** Conceptualization, Writing - original draft. **Kaitai Liang:** Methodology, Writing - review & editing, Writing - original draft. **Jianting Ning:** Writing - review & editing, Writing - original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

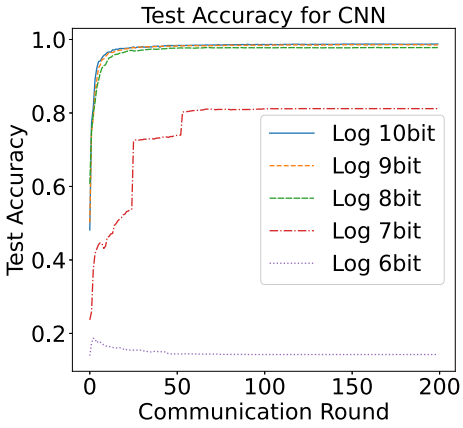
This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 61972094, 62032005), in part by the Young Talent Promotion project of Fujian Science and Technology Association, and in part by the Science Foundation of Fujian Provincial Science and Technology Agency (2020J02016).

Appendix A. Security definitions

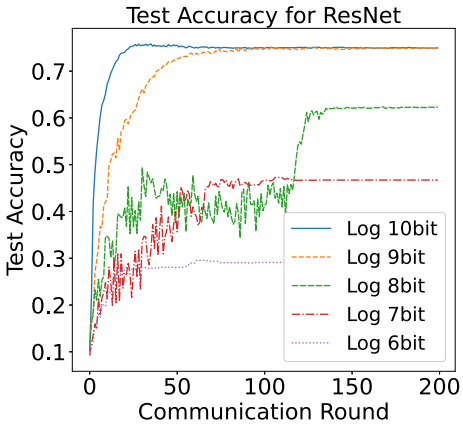
Definition 1 (Discrete Logarithm Hard Problem – DLHP) Discrete Logarithm is considered to be hard if

$$Pr\left[DLog_{\mathcal{G},\mathcal{A}}(\alpha) = 1\right] \leqslant negl(\alpha)$$

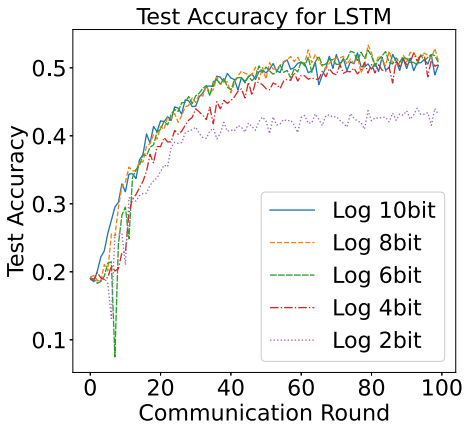
Definition 2 (Computational Diffle-Hellman – CDH) CDH is considered to be hard if



(a) CNN for MNIST dataset



(b) ResNet for CIFAR10 dataset



(c) LSTM for Shakespeare dataset

Fig. 10. The test accuracy of the global model with different encoding lengths.

Table 5
Model test performance over different brute force recovery time on three datasets.

Encoding bit	6bit	7bit	8bit	9bit	10bit	11bit	12bit	13bit	14bit	15bit
CIFAR-10										
Accuracy (%)	29.1	46.71	62.4	74.87	74.96	76.03	75.31	75.68	75.54	75.89
Average Time (s)	\	\	\	\	\	0.1006	0.2508	0.5360	1.2207	2.6491
Maximum Time (s)	\	\	\	\	\	1.4608	3.9389	7.5755	16.1956	42.2203
MNIST										
Accuracy (%)	14.25	81.18	97.79	98.53	98.75	98.97	98.91	98.92	98.98	99
Average Time (s)	\	\	\	\	\	0.1267	0.2312	0.5585	1.1146	1.9786
Maximum Time (s)	\	\	\	\	\	1.2942	1.6247	6.3499	9.0804	18.0468
Shakespeare										
Accuracy (%)	50.80	516.65	51.04	50.11	50.33	50.42	51.67	51.15	50.47	51.26
Average Time (s)	\	\	\	\	\	1.4180	2.5906	5.4531	10.8060	23.8876
Maximum Time (s)	\	\	\	\	\	3.2861	7.0456	11.9095	31.8851	55.2088

$$Pr\left[\mathscr{A}(g,g^a,g^b)=(g^{ab})\right]\leqslant negl(\alpha)$$

Definition 3 (Decisional Diffie-Hellman – DDH) DDH is considered to be hard if

$$|Pr\left[\mathscr{A}(g,g^a,g^b,g^c)=1\right]-Pr\left[\mathscr{A}(g,g^a,g^b,g^{ab})=1\right]|\leqslant negl(\alpha)$$

Definition 4 Indistinguishability – Chosen plaintext attacks (IND - CPA).

Consider a following game between an adversary $\mathscr{A}$ and a challenger:

Set up: Challenger generates public parameters $<g,p,q>$, public key h and secret key s, then sends public parameters and pk to the $\mathscr{A}$.

Challenge: $\mathscr{A}$ chooses two plaintexts m_0,m_1 with same length, then sends them to the challenger. Challenger randomly selects $b\in\{0,1\}$, encrypts m_b

$$C=Enc(pk,m_b)$$

and sends challenge ciphertext C to $\mathscr{A}$. Finally, $\mathscr{A}$ would compute b'.

A scheme is security against CPA if the advantage

$$\mathbf{Adv}_{\mathscr{A}}^{CPA}(\alpha)=\left|Pr(b=b')-\frac{1}{2}\right|$$

is negligible.

Appendix B. Security analysis

- Parameters generation: server runs a polynomial-time $\mathscr{G}(1^{\alpha})$ to generate public parameters $<\mathbb{G},g,p,q>$, where g is a generator of $\mathbb{G}$ which is a cyclic group with prime order q,p is a large prime number satisfying $q|p-1,y$ is a random element in $\mathbb{G}$ and α corresponding to security level is bit length of q. Given a polynomial-time algorithm $\mathscr{A}$ and a negligible function $negl$, the security of key generation and encryption are based on following definitions.
- Key generation: $x\leftarrow\$\mathbb{Z}_q^*, h=g^x(\mathrm{mod}p)$.
- Encryption: $g^m\in\mathbb{Z}_p^*, r\leftarrow\$\mathbb{Z}_q^*, <c_1=g^r(\mathrm{mod}p), c_2=g^m\cdot h^r(\mathrm{mod}p)>$.
- Decryption: $\frac{c_2}{c_1^x}=\frac{mh^r}{(g^r)^x}=\frac{g^m\cdot g^{xr}}{g^{rx}}=g^m(\mathrm{mod}p)$

Additive ElGamal is CPA-secure against chosen ciphertext attacks if the advantage of $\mathscr{A}$ is negligible in α satisfying as follows:

$$\mathbf{Adv}_{\mathscr{A}}^{CPA}(\alpha)=\left|Pr(b=b')-\frac{1}{2}\right|$$

Proof. Ciphertexts under additive ElGamal:

$$C=(c_1,c_2)=\left(g^r,g^{m_b}\cdot h^r\right)=g^{m_b+x\cdot r}$$

$\mathscr{A}$ computes:

$$C'=C\cdot g^{m_{b'}^{-1}}=g^{x\cdot r}\ \ iff\ \ b=b'$$

Besides, $\mathscr{A}$ wants to determine whether C' is DH agreement. $\mathscr{A}$ could "win" this game with a probability:

$$\mathbf{Adv}_{\mathscr{A}}^{ElG}(\alpha)=||\ \ Pr[\mathscr{A}(g,g^x,g^r,g^{x\cdot r+m_b-m_{b'}})=1]-Pr[\mathscr{A}(g,g^x,g^r,g^{x\cdot r})=1]|-\tfrac{1}{2}|\leqslant negl(\alpha)$$

Theorem 1 The exponential ElGamal and distribute key generation are CPA-secure and correctness-satisfying according to [13,24]. □

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org/. software available from tensorflow.org. .

[2] Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L., Deep learning with differential privacy, in: The 2016 ACM CCS. .

[3] Agarap, A.F., 2018. Deep learning using rectified linear units (relu). CoRR abs/1803.08375. url:http://arxiv.org/abs/1803.08375, arXiv:1803.08375. .

[4] Amiri, M.M., Gunduz, D., Kulkarni, S.R., Poor, H.V., 2020. Federated learning with quantized global model updates. arXiv preprint arXiv:2006.10672. .

[5] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, Recommendation for key management – part 1: General (revision 3), NIST Special Publication Revision 3 (2005).

[6] J.P. Berrut, L.N. Trefethen, Barycentric lagrange interpolation, SIAM Rev. 46 (2004) 501–517.

[7] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K., 2017. Practical secure aggregation for privacy-preserving machine learning. .

[8] D. Boneh, A. Sahai, B. Waters, Functional encryption: Definitions and challenges, Theory of Cryptography Conference, Springer (2011) 253–273.

[9] Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D., 2013. Classical hardness of learning with errors, in: Proceedings of the forty-fifth annual ACM symposium on Theory of computing, pp. 575–584. .

[10] Caldas, S., Wu, P., Li, T., Konecný, J., McMahan, H.B., Smith, V., Talwalkar, A., 2018. LEAF: A benchmark for federated settings. CoRR abs/1812.01097. url: http://arxiv.org/abs/1812.01097, arXiv:1812.01097. .

[11] Cao, X., Jia, J., Gong, N.Z., 2021. Provably secure federated learning against malicious clients. arXiv preprint arXiv:2102.01854. .

[12] Y. Chen, X. Sun, Y. Jin, Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation, IEEE Trans. Neural Networks Learn. Syst. 31 (2020) 4229–4238, https://doi.org/10.1109/TNNLS.2019.2953131.

[13] R. Cramer, R. Gennaro, B. Schoenmakers, A secure and optimally efficient multi-authority election scheme, Eur. Trans. Telecommun. 8 (1997) 481–490.

[14] Dai, X., Yan, X., Zhou, K., Ng, K.K., Cheng, J., Fan, Y., 2019. Hyper-sphere quantization: Communication-efficient sgd for federated learning. arXiv preprint arXiv:1911.04655. .

[15] Damgård, I., Jurik, M., 2001. A generalisation, a simpli. cation and some applications of paillier's probabilistic public-key system, in: PKC, Springer. .

[16] L. Deng, D. Yu, Deep learning: methods and applications, Found. Trends Signal Process. 7 (2014) 197–387.

[17] Y. Du, S. Yang, K. Huang, High-dimensional stochastic gradient quantization for communication-efficient edge learning, IEEE Trans. Signal Process. 68 (2020) 2128–2142.

[18] C. Dwork, Differential privacy: a survey of results, in: M. Agrawal, D. Du, Z. Duan, A. Li (Eds.), Theory and Applications of Models of Computation, Springer, Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 1–19.

[19] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Inf. Theory 31 (1985) 469–472.

[20] Feldman, P., A practical scheme for non-interactive verifiable secret sharing, in: SFCS 1987, IEEE. .

[21] Fredrikson, M., Jha, S., Ristenpart, T., Model inversion attacks that exploit confidence information and basic countermeasures, in: the 22nd ACM CCS. .

[22] Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., Ristenpart, T., 2014. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing, in: 23rd {USENIX} Security Symposium {USENIX} Security 14), pp. 17–32. .

[23] Geiping, J., Bauermeister, H., Dröge, H., Moeller, M., 2020. Inverting gradients – how easy is it to break privacy in federated learning? arXiv:2003.14053. .

[24] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T., 1999. Secure distributed key generation for discrete-log based cryptosystems, in: Eurocrypt, Springer. .

[25] C. Gentry, D. Boneh, A fully homomorphic encryption scheme, vol. 20, Stanford University Stanford, 2009.

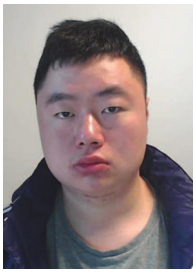
[26] Geyer, R.C., Klein, T., Nabi, M., 2017. Differentially private federated learning: A client level perspective. arXiv preprint arXiv:1712.07557. .

[27] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep learning, vol. 1, MIT Press, Cambridge, 2016.

[28] Hao, M., Li, H., Xu, G., Liu, S., Yang, H., Towards efficient and privacy-preserving federated deep learning, in: ICC 2019, IEEE. .

[29] B. Hitaj, G. Ateniese, F. Perez-Cruz, Deep models under the gan: information leakage from collaborative deep learning, in: Proceedings of the 2017 ACM

SIGSAC Conference on Computer and Communications Security, 2017, pp. 603–618.
[30] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.
[31] Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167. url:http://arxiv.org/abs/1502.03167, arXiv:1502.03167. .
[32] M. Kim, J. Lee, L. Ohno-Machado, X. Jiang, Secure and differentially private logistic regression for horizontally distributed data, IEEE Trans. Inf. Forensics Secur. 15 (2020) 695–710, https://doi.org/10.1109/TIFS.2019.2925496.
[33] F. Knirsch, A. Unterweger, M. Unterrainer, D. Engel, Comparison of the Paillier and ElGamal Cryptosystems for Smart Grid Aggregation Protocols, in: Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP), SciTePress, Valetta, Malta, 2020, pp. 232–239.
[34] Krizhevsky, A., Nair, V., Hinton, G., Cifar-10 (canadian institute for advanced research) URL: http://www.cs.toronto.edu/kriz/cifar.html. .
[35] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (2017) 84–90.
[36] Kursawe, K., Danezis, G., Kohlweiss, M., 2011. Privacy-friendly aggregation for the smart-grid. .
[37] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[38] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, Handbook Brain Theory Neural Networks 3361 (1995) 1995.
[39] LeCun, Y., Cortes, C., 2010. MNIST handwritten digit database URL: http://yann.lecun.com/exdb/mnist/. .
[40] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption, Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer (2010) 62–91.
[41] H. Li, T. Han, An end-to-end encrypted neural network for gradient updates transmission in federated learning, in: 2019 Data Compression Conference (DCC), 2019, https://doi.org/10.1109/DCC.2019.00101, 589–589.
[42] L. Li, Y. Fan, M. Tse, K.Y. Lin, A review of applications in federated learning, Comput. Ind. Eng. 149 (2020) , https://doi.org/10.1016/j.cie.2020.106854 106854.
[43] Li, S., Cheng, Y., Wang, W., Liu, Y., Chen, T., 2020b. Learning to detect malicious clients for robust federated learning. arXiv preprint arXiv:2002.00211. .
[44] Y. Lu, M. Zhu, Privacy preserving distributed optimization using homomorphic encryption, Automatica 96 (2018) 314–325, https://doi.org/10.1016/j.automatica.2018.07.005.
[45] X. Ma, F. Zhang, X. Chen, J. Shen, Privacy preserving multi-party computation delegation for deep learning in cloud computing, Inf. Sci. 459 (2018) 103–116.
[46] Mandal, K., Gong, G., Privfl: Practical privacy-preserving federated regressions on high-dimensional data over mobile networks, in: The 2019 ACM CCSW. .
[47] S. Mao, Y. Tang, Z. Dong, K. Meng, Z.Y. Dong, F. Qian, A privacy preserving distributed optimization algorithm for economic dispatch over time-varying directed networks, IEEE Trans. Ind. Inf. 17 (2021) 1689–1701, https://doi.org/10.1109/TII.2020.2996198.
[48] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A., 2017. Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR. pp. 1273–1282. .
[49] M.F. Møller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks 6 (1993) 525–533.
[50] M.A. Nielsen, Neural networks and deep learning, vol. 2018, Determination press San Francisco, CA, 2015.
[51] Orekondy, T., Oh, S.J., Zhang, Y., Schiele, B., Fritz, M., 2018. Gradient-leaks: Understanding and controlling deanonymization in federated learning. arXiv preprint arXiv:1805.05838. .
[52] Paillier, P., 1999. Public-key cryptosystems based on composite degree residuosity classes, in: TAMC, Springer. .
[53] Pedersen, T.P., 1991. Non-interactive and information-theoretic secure verifiable secret sharing, in: CRYTO, Springer. .
[54] L.T. Phong, Y. Aono, T. Hayashi, L. Wang, S. Moriai, Privacy-preserving deep learning via additively homomorphic encryption, IEEE Trans. Inf. Forensics Secur. 13 (2018) 1333–1345, https://doi.org/10.1109/TIFS.2017.2787987.
[55] O. Regev, The learning with errors problem, Invited survey in CCC 7 (2010) 11.
[56] Ribero, M., Vikalo, H., 2020. Communication-efficient federated learning via optimal client sampling. arXiv:2007.15197v2. .
[57] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Networks 61 (2015) 85–117.
[58] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (1997) 2673–2681.
[59] Shakespeare, W., The complete works of william shakespeare. .
[60] A. Shamir, How to share a secret, Commun. ACM 22 (1979) 612–613.
[61] Shokri, R., Shmatikov, V., Privacy-preserving deep learning, in: the 22nd ACM CCS. .
[62] M.A. Tanner, W.H. Wong, The calculation of posterior distributions by data augmentation, J. Am. Stat. Assoc. 82 (1987) 528–540.
[63] Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., Zhou, Y., A hybrid approach to privacy-preserving federated learning, in: the 12th ACM AISec, Association for Computing Machinery, New York, NY, USA. url: https://doi.org/10.1145/3338501.3357370, doi:10.1145/3338501.3357370. .
[64] Uspensky, J.V., 1937. Introduction to mathematical probability. .
[65] D.A. Van Dyk, X.L. Meng, The art of data augmentation, J. Comput. Graphical Stat. 10 (2001) 1–50.
[66] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, H. Qi, 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.
[67] K. Wei, J. Li, M. Ding, C. Ma, H.H. Yang, F. Farokhi, S. Jin, T.Q. Quek, H.V. Poor, Federated learning with differential privacy: algorithms and performance analysis, IEEE Trans. Inf. Forensics Secur. (2020).
[68] Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., Li, H., 2017. Terngrad: Ternary gradients to reduce communication in distributed deep learning, in: NIPS. .
[69] J. Xu, W. Du, Y. Jin, W. He, R. Cheng, Ternary compression for communication-efficient federated learning, IEEE Trans. Neural Networks Learn. Syst. (2020).
[70] Xu, J., Jin, Y., Du, W., Gu, S., 2021. A federated data-driven evolutionary algorithm. arXiv preprint arXiv:2102.08288. .
[71] Xu, R., Baracaldo, N., Zhou, Y., Anwar, A., Ludwig, H., 2019. Hybridalpha: an efficient approach for privacy-preserving federated learning, in: The 12th ACM AISec, Association for Computing Machinery, New York, NY, USA. doi:10.1145/3338501.3357371. .
[72] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2019) 1–19.
[73] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, K.H. Johansson, A survey of distributed optimization, Annu. Rev. Control 47 (2019) 278–305, https://doi.org/10.1016/j.arcontrol.2019.05.006.
[74] Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y., Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning, in: 2020 USENIX ATC. URL: https://www.usenix.org/conference/atc20/presentation/zhang-chengliang. .
[75] X. Zhang, J. Liu, Z. Zhu, E.S. Bentley, Communication-efficient network-distributed optimization with differential-coded compressors, in: IEEE INFOCOM 2020 – IEEE Conference on Computer Communications, 2020, pp. 317–326, https://doi.org/10.1109/INFOCOM41043.2020.9155432.
[76] Zhao, B., Mopuri, K.R., Bilen, H., 2020. idlg: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610. .
[77] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, Y. Chen, Privacy-preserving collaborative deep learning with unreliable participants, IEEE Trans. Inf. Forensics Secur. 15 (2020) 1486–1500, https://doi.org/10.1109/TIFS.2019.2939713.
[78] Zhu, C., Han, S., Mao, H., Dally, W.J., 2016. Trained ternary quantization. arXiv preprint arXiv:1612.01064. .
[79] Zhu, H., Jin, Y., 2020. Real-time federated evolutionary neural architecture search. arXiv preprint arXiv:2003.02793. .
[80] H. Zhu, H. Zhang, Y. Jin, From federated learning to federated neural architecture search: a survey, Complex Intell. Syst. 7 (2021) 639–657, url: arXiv preprint arXiv:2009.05868.
[81] Zhu, L., Liu, Z., Han, S., 2019. Deep leakage from gradients. arXiv:1906.08935. .



Hangyu Zhu received B.Sc. and M.Sc. from Yangzhou University, China in 2015, and the RMIT University, Australia in 2017, respectively. He is currently a PhD student with the Department of Computer Science, University of Surrey, working on evolutionary neural architecture search for federated learning.

Rui Wang received the B.Sc. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2017, and the M.Sc. degree from University of Southampton, U.K., in 2018. He is currently pursuing the Ph.D. degree, focusing on Privacy-preserving machine learning, with the Department of Intelligent Systems, Delft University of Technology, Delft, Netherlands.

Yaochu Jin is currently a Distinguished Chair Professor in Computational Intelligence, Department of Computer Science, University of Surrey, Guildford, U.K. He was a Finland Distinguished Professor awarded by the Finnish Funding Agency for Innovation, Finland, and Changjiang Distinguished Visiting Professor by the Ministry of Education, China. Most recently, he was awarded the Alexander von Humboldt Professorship for Artificial Intelligence by the Federal Ministry of Education and Research, Germany. His research interests lie in the cross-disciplinary areas of computational intelligence, computational neuroscience and computational biology. Prof Jin is presently the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS and Complex & Intelligent Systems. He is an IEEE Distinguished Lecturer (2017–2019) and was the Vice President for Technical Activities of the IEEE Computational Intelligence Society (2014–2015). He is the recipient of the 2015, 2017, and 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award, and the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award. He is a Member of Academia Europaea and Fellow of IEEE.

Kaitai Liang joined the Delft University of Technology, the Netherlands in 2020. Before that he was an assistant professor in secure systems at the Department of Computer Science, University of Surrey. He received his Ph.D. degree from the Department of Computer Science, City University of Hong Kong. His research interests are applied cryptography and information security; in particular, data encryption, blockchain security, post-quantum crypto, privacy enhancing technology, and privacy-preserving machine learning.

Jianting Ning received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University in 2016. He is currently a research fellow at School of Information Systems, Singapore Management University. He is a Professor with the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, China. Previously, he was a research fellow at Department of Computer Science, National University of Singapore. His research interests include applied cryptography and information security, in particular, public key encryption, secure and privacy-preserving computation.