2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid) | 978-1-7281-9586-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCGrid51090.2021.00101

Partially Encrypted Multi-Party Computation for Federated Learning

Ekanut Sotthiwat*†, Liangli Zhen†, Zengxiang Li‡, Chi Zhang†
*National University of Singapore, Singapore
Email: e0431608@u.nus.edu
†Institute of High Performance Computing, A*STAR, Singapore
Email: {zhen_liangli, zhang_chi}@ihpc.a-star.edu.sg
‡ENNEW Digital Research Institute, ENN Group, Beijing, China
Email: zengxiang_li@outlook.com

Abstract—**Multi-party computation (MPC) allows distributed machine learning to be performed in a privacy-preserving manner so that end-hosts are unaware of the true models on the clients. However, the standard MPC algorithm also triggers additional communication and computation costs, due to those expensive cryptography operations and protocols. In this paper, instead of applying heavy MPC over the entire local models for secure model aggregation, we propose to encrypt critical part of model (gradients) parameters to reduce communication cost, while maintaining MPC's advantages on privacy-preserving without sacrificing accuracy of the learnt joint model. Theoretical analysis and experimental results are provided to verify that our proposed method could prevent deep leakage from gradients attacks from reconstructing original data of individual participants. Experiments using deep learning models over the MNIST and CIFAR-10 datasets empirically demonstrate that our proposed partially encrypted MPC method can reduce the communication and computation cost significantly when compared with conventional MPC, and it achieves as high accuracy as traditional distributed learning which aggregates local models using plain text.**

Index Terms—**distributed machine learning, privacy-preserving learning, federated learning, multi-party computation**

1. Introduction

Machine learning, especially deep learning, has made significant breakthroughs in many domains of science, business and government, such as manufacturing, transportation, finance, and healthcare [1, 2]. The centralised learning mainly contributes to these remarkable successes on large-scale datasets. With the popularity of modern technologies of edge computing [3] and the Internet of Things [4, 5], machine learning has witnessed a dramatic change in the way it computes. Data in many real-world scenarios are naturally distributed and owned by different organisations/users. Due to the competition of different organisations, data privacy security, and administrative regulations, it is almost impossible to upload the data across countries and institutions for centralised learning [2, 6].

To leverage the distributed confidential datasets, the researchers in Google proposed the concept of federated learning [7]. Federated learning enables multiple participants to build a joint machine learning model collaboratively without sharing data, *i.e.*, no need to upload private data to a centralised server or to exchange data across participants, thus allowing to address the critical issues such as data privacy and data security. However, there is a potential data leakage in this framework; even the private data stay in the local participant. Four majors types of information may be leaked from federated learning: 1) membership leakage [8], 2) unintended feature leakage [9], 3) the representative of the class of original data leakage [10], and 4) the original data leakage [11, 12]. The last type of data leakage is the most unacceptable for privacy-sensitive participants. Previous studies have shown a possible way of reconstructing the original data of the private training datasets using the algorithms of deep leakage from gradients [11, 12]. In this case, one natural question would be how to develop a privacy-preserving distributed machine learning approach, while avoiding potential information leakage from gradients during transmission and aggregation?

Cryptography-based methods have attracted the most interest as they are designed to handle data privacy, integrity and authentication issues by transforming information from a readable state to almost nonsense. Specifically, Secure Multi-Party Computation (MPC) [13] provides a generic primitive that enables distributed parties to jointly compute an arbitrary functionality without revealing their private inputs and outputs [14], hence naturally suitable for privacy-preserving model aggregation amongst trust-less parties. It addresses the problem of cooperative computation in a secure fashion so that local models are split and transformed into multiple pieces of secret shares. The aggregated model is calculated by exchanging these secrete shares amongst participants, following well-designed MPC protocols. As a result, no single party has sufficient pieces of secret shares to reconstruct any specific local model and therefore protect local models from leaking to certain parties, and the aggregated server is applicable.

Compared with other cryptography-based methods, such as Differential Privacy (DP) [15, 16] and Homomorphic Encryption (HE) [17], MPC-based distributed learning [13, 18–22] has several advantages. First of all, the MPC method provides an almost accurate aggregation of the local models and eliminates the trade-off between data privacy and model performance. In contrast, DP-based distributed learning generally need to consider a case-by-case noise adding strategy, and the performance of the global model relies on the introduced noise level. Secondly, MPC protocol prevents each party from reconstructing private data using secret shares, only if it could collude with all (for additive secret share MPC protocol [23]) or a certain number (for Shamir secret share MPC protocol [24]) of other local participants [1]. In contract, HE-based distributed learning generally need to assume computation peers or centralised coordinators are trusted, which may not exist in real scenarios.

However, the aforementioned benefits do not come as a free dinner. On the other side of the coin is the fact that additive secrete share MPC protocol [23] generally requires each participant to generate m secret shares of the local model and distribute them to all computation peers, where m represents the total number of distributed learning participants. Considering the number of potential participants is large in most cases, standard MPC will inevitably lead to massive local computation and communication costs when secrete shares are generated, transmitted and calculated among participants.

In this paper, instead of applying MPC over the entire local models for secure aggregation, we propose a partially encrypted MPC solution by encrypting critical parts of model parameters (gradients) that are vulnerable to privacy-preserving attacks. Specifically, only the first layer of local models is encrypted with MPC strategy, while the rest are sent directly to the centralised node. Such a solution significantly reduces the extra computation and communication overhead led by MPC, while inheriting MPC benefits in both privacy-preserving and model accuracy perspectives. We perform a series of experiments to verify that the proposed solution achieves the merits of high prediction accuracy (as Non-MPC counterpart), low communication and computation costs, and high effectiveness of defending attacks of deep leakage from gradients (DLG) [11] and the improved DLG (iDLG) [12].

2. Related Work

MPC has been widely adopted in federated learning frameworks for privacy-preserving model aggregation due to the aforementioned advantages. CrypTen and OpenMined enable MPC-based federated learning for neural network models built on PyTorch. However, they provide a very limited study on MPC overhead and performance evaluation on the entire system. Google provides computation and communication complexity analysis on its practical MPC protocol and experimental evaluation over a large number of mobile devices [25]. A two-phase MPC-enabled Federated Learning framework was proposed in [26] to reduce communication cost and improve system scalability via electing no-colluding committee members for conducting secure model aggregation. Different from the above efforts, this paper looks into the network structure of complex deep learning models and then apply privacy-preserving MPC-based model aggregation over critical neural network layers only.

Information leakage from gradients [11, 27, 28] has a significant impact on the way distributed machine learning works, especially for collaborative learning with multiple parties. In classical distributed machine learning, participants generally trust the server and send their local models as plain text to the server for aggregation. However, when the server becomes a curious one, such a direct information exchange step leaves the possibilities of reconstructing the original images on the server. Lyn *et al.* [29] provided a survey on threat models as well as poisoning and inference attacks on Federated Learning.

To give a simple example, we consider the framework proposed in [11]. It assumes that the local participants follow the standard protocol rules and transfer the gradients as plain text to the server. After receiving the gradient from the local clients, the server initialises the input (x) and the labels (y). The malicious server computes the dummy gradients from dummy inputs (x, y) and calculates the difference between dummy gradients and gradients calculated by a victim. Next, the server updates the dummy inputs and labels by the standard gradient-based method and stops when the dummy gradient equal to the actual gradient. As a result, when the batch size is one, the attacker can reconstruct the images that are the same as or very similar to the ground truth images [10]. However, the convergence speed of data extraction in DLG is slow, and it cannot constantly obtain accurate ground-truth labels. Zhao *et al.* [12] improved the DLG method to steal the data and the corresponding labels from the shared gradients in a distributed learning system.

3. The Proposed Method

3.1. Problem Formulation

In federated learning, we aim to learn a joint model $\mathcal{M}_f$ on data $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_m\}$ that are located on m respective distributed local nodes or devices $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m\}$. The data will not be allowed to leave the local nodes during the model learning and the inference due to data privacy-preserving considerations. However, the inference accuracy of the model $\mathcal{M}_f$ is desired to be very close to that of the model $\mathcal{M}_s$ that trained on the combination of all data in $\mathcal{X}$. Furthermore, the data on different nodes may have distinctive distributions, which makes federated learning more challenging.

3.2. Partially Encrypted MPC-Based Federated Learning System

In this work, we propose a partially encrypted MPC (PEMPC) approach to encrypt the weights and train the

1. This paper focuses on additive secret share MPC protocol only.

joint model, as shown in Figure 1. In the system, the m participants have the same neural network architecture collaboratively learn the joint model with the help of a model aggregation server. A typical assumption is that the participants are honest, whereas the server is honest-but-curious [2]. To prevent information leakage from the participants to the server, we do not send the whole model of any participant to the server. Specifically, we split the gradients of the local model, $\mathcal{G}_i$, into two types, which are associated with the dash lines (Type A, $\mathcal{G}_i^A$) and the solid lines (Type B, $\mathcal{G}_i^B$) in the network, respectively. Accordingly, the weight parameters of the model are split into Type A ($\mathcal{W}_i^A$) and Type B ($\mathcal{W}_i^B$).

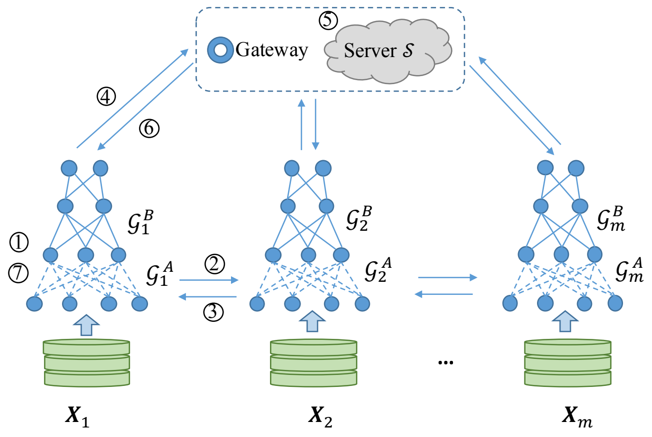


Figure 1. Framework for the partially encrypted MPC-based distributed machine learning. In each network, the Type A gradients $\mathcal{G}_i^A$ are associated with the dash lines and the Type B gradients $\mathcal{G}_i^B$ are associated with the solid lines. Only the Type A gradients will be aggregated using the secure MPC for encryption.

The training process of our proposed partially encrypted MPC-based distributed machine learning contains the following main steps:

- Step 1: each participant $\mathcal{P}_i$ computes the gradients of the model $\mathcal{M}_f$ locally based on its own dataset $\mathbf{X}_i$, denoted as $\mathcal{G}_i$.
- Step 2: each participant $\mathcal{P}_i$ generates and sends the secret shares of the Type A gradients $\mathcal{G}_i^A$ to other local participants.
- Step 3: each participant $\mathcal{P}_j$ receives the secret shares of the Type A gradients $\mathcal{G}_1^A, \mathcal{G}_2^A, \ldots, \mathcal{G}_m^A$ from other local participants, and $\mathcal{P}_j$ sums the secret shares up to replace the values of $\mathcal{G}_j^A$.
- Step 4: each participant $\mathcal{P}_i$ sends the Type A gradients $\mathcal{G}_i^A$ and the Type B gradients $\mathcal{G}_i^B$ to the server.
- Step 5: the server conducts the aggregation of the gradients from the local participants as

$$\mathcal{G}^A = \frac{1}{m}\mathcal{G}_i^A \tag{1}$$

and

$$\mathcal{G}^B = \frac{1}{m}\mathcal{G}_i^B. \tag{2}$$

- Step 6: the server sends back the aggregated results to the participants.
- Step 7: each participant $\mathcal{P}_i$ updates the weight parameters of the model as

$$\mathcal{W} = \mathcal{W} + \alpha\mathcal{G}, \tag{3}$$

where α is the learning rate.

The above steps will continue iteratively until the termination condition is reached, and completing the entire training process. The termination condition can be the maximum number of training epochs is reached, the loss function converges, or other user-defined conditions. The neural network architecture in our framework can be any one of deep neural network architectures, such as LeNet-5 [30], VGG-16 [31], ResNet-152 [32], and DenseNet-121 [33].

3.3. Partially Encrypted MPC Prevents Recovery of Training Data

The adversarial attack approaches in [11] and [12] have shown that the private training data may be obtained based on the publicly shared gradients. As introduced previously that the server may be honest-but-curious [2]. The server can recover the original training set of all the participants using the approaches like deep leakage from gradient (DLG) [11] and its improved version iDLG [12]. We observe that the gradients of the first hidden layer of $\mathcal{P}_i$ are calculated based on the input data $\mathbf{X}_i$ and the back-propagation results from the second hidden layer. Thus, we encrypted the first hidden layer as the gradient type A to prevent the reconstructing of the training data.

Denoting the weight parameters of the first hidden layer as $\mathbf{W} \in \mathbb{R}^{d\times n}$ and the bias as $\mathbf{b}$, we can obtain the output of the first hidden layer for an input data point $\mathbf{x}$ as

$$\mathbf{y} = f(\mathbf{z}); \quad \mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}, \tag{4}$$

where $f(\cdot)$ is the activation function and typically sets as the sigmoid, ReLu, or softmax function.

By using DLG or iDLG, the server can recover the accurate results of $\mathbf{y}$ if the participant only encrypts the weight parameters of the first hidden layer. Then, it can also obtain $\mathbf{z}$ accurately if $f(\cdot)$ is absolutely monotonic. The problem is now transformed to recover $\mathbf{x}$ in Equation (4) under the condition that only $\mathbf{z}$ is known, which is a blind source separation problem [34, 35].

Assumption 1. *For each input vector $\mathbf{x}$, there are more than n non-zero elements.*

Under the assumption 1, we have the conclusion:

Lemma 1. *There exist an infinite number of solutions for $\mathbf{x}$ in the linear system $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$ if $d > n$.*

It provides a theoretical foundation to guarantee that the attacker cannot recover the input vector when the dimensionality of the input data is more than the number of neurons in the first hidden layer.

3.4. Secure Multi-Party Computation

We will illustrate how to use secure MPC to aggregate the Type A gradients $\mathcal{G}_1^A, \mathcal{G}_2^A, \ldots, \mathcal{G}_m^A$. Secure MPC is one of the cryptography techniques, which aims to create methods for participants to jointly compute a function over their inputs while keeping those inputs private. A typical secret sharing MPC protocol is as shown in Figure 2.

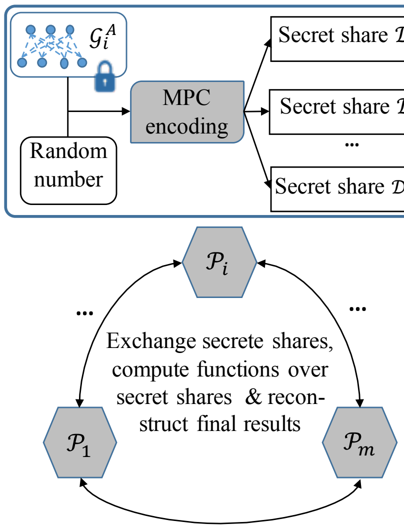

$\mathcal{G}_i^A$
Secret share $\mathcal{D}_{i1}^A$
MPC encoding
Secret share $\mathcal{D}_{i2}^A$
...
Random number
Secret share $\mathcal{D}_{im}^A$
$\mathcal{P}_i$
...
...
Exchange secrete shares, compute functions over secret shares & recon-struct final results
$\mathcal{P}_1$
$\mathcal{P}_m$

Figure 2. Secure multi-party computation.

At Step 2 of our method, the matrix of Type A gradients of each participant, $\mathcal{G}_i^A$, is split into $(m-1)$ secret shares $\mathcal{D}_{i1}^A, \mathcal{D}_{i2}^A, \ldots, \mathcal{D}_{i(i-1)}^A, \mathcal{D}_{i(i+1)}^A, \ldots, \mathcal{D}_{im}^A$, which in combination yield the original $\mathcal{G}_i^A$. In this work, we consider the additive secret sharing MPC protocol. At Step 3 of our method, $\mathcal{P}_j$ receives the secret shares of Type A gradients as $\{\mathcal{D}_{ij}^A\}_{i=1,i\neq j}^m$ and computes the initial result for the model aggregation as

$$\mathcal{D}_j^A = \sum_{i=1,i\neq j}^{m} \mathcal{D}_{ij}^A, \tag{5}$$

and replaces the values of $\mathcal{G}_j^A$ as

$$\mathcal{G}_j^A = \mathcal{D}_j^A. \tag{6}$$

At Step 5 of our method, the server will conduct the aggregation of the gradients based on the aggregated secret shares, *i.e.*, $\mathcal{G}_1^A, \mathcal{G}_2^A, \ldots, \mathcal{G}_m^A$, to obtain the final aggregated result as Equation (1). Note that the MPC result is obtained by conducting two stages of secret shares sum up at Step 3 and Step 5, respectively.

4. Experimental Study

The idea of the partially encrypted MPC strategy provides an alternative solution for encrypted peer-to-peer communication. By design, it can protect local participants' privacy by transforming local gradients into secrete shares (integers) while at the same time enjoys the merits of *high accuracy, low communication and low computation requirements*. These properties of partial MPC shall be validated in this section by performing experiments on a system with 10 local participants and 1 central server.

4.1. Comparison of Prediction Accuracy

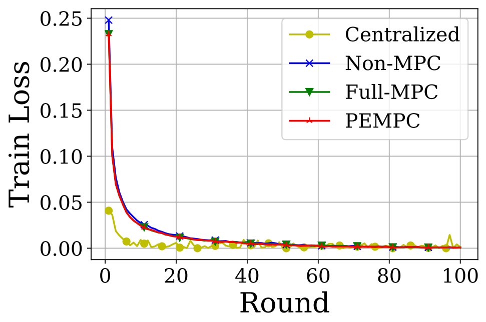
One of the key advantages of the standard MPC algorithm is that it provides strong model protections while minimally triggers accuracy drops. Therefore, before proceeding to the communication reduction, the proposed partial encrypted MPC algorithm is firstly expected to inherit such merits and obtain high accuracy in practice. To validate this advantage, we apply LeNet-5 architecture [30] to classify the images in the MNIST dataset. We investigate the loss and accuracy curves during training and testing. The results of the two different encryption strategies, namely MPC and PEMPC, are shown in Figure 3. For better illustration, we also plot the learning curves of standard distributed optimisation method without any encryption (denoted as "Non-MPC") and centralised learning, which act as our baselines. The initialisation of the model, the optimisation method and the learning rate are all set to be the same for fairness.

Results in Figure 3(a) show that the distributed learning methods, *i.e.*, Non PMC, MPC, and PEMPC, share a similar convergence speed in the training phase, but the speed is slightly slower than the centralised training. From the results in Figure 3(b), we can see that the accuracy of the MPC-based strategies are almost identical to the Non-MPC baseline (blue line), which demonstrates that the extra MPC encryption step does not trigger accuracy drops during the learning. The underlying reason is that the MPC methods provide an unbiased aggregation for local models and eliminate the trade-off between data privacy and model performance. This is in contrast to other privacy-preserving methods (e.g., differential privacy) that often suffer from accuracy drops after adding noises to local gradients or weights. Similar patterns can be observed for VGG-16 [31] on the CIFAR-10 dataset [36], see Figure 3(c) and Figure 3(d).

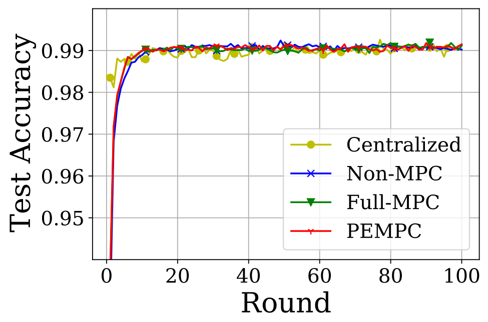
To summarise, the MPC methods allow the distributed machine learning to be performed in a secure way while not incurring a loss. Note this conclusion is true for both standard MPC and PEMPC, which represents an ideal situation where the proposed PEMPC method can protect the participants' privacy without sacrificing the overall learning performance.

4.2. Communication and Computation Cost

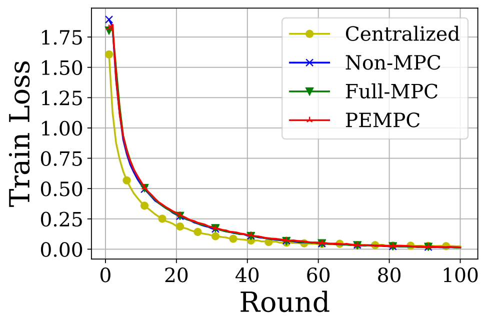
As stated earlier, the secure updating scheme of standard MPC does not come as a free dinner. One of the major issues of the full MPC method is the extra communication cost led by peer-to-peer model transmission. In general, each local participant has first to split its model into m parts and send $(m-1)$ fractions to the rest participants and then transmits the aggregated model to the server. In other



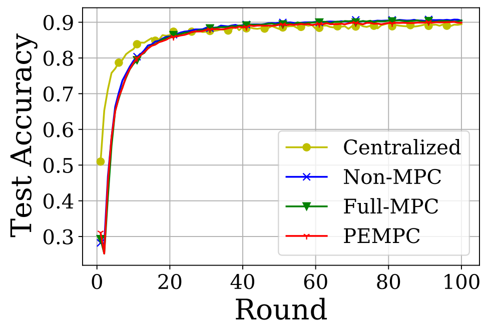
(a) Train loss on the MNIST dataset.



(b) Test accuracy on the MNIST dataset.



(c) Train loss on the CIFAR-10 dataset.



(d) Test accuracy on the CIFAR-10 dataset.

Figure 3. Performance of LeNet-5 on the MNIST dataset and VGG-16 on the CIFAR-10 dataset. Samples are randomly distributed to the local participants.

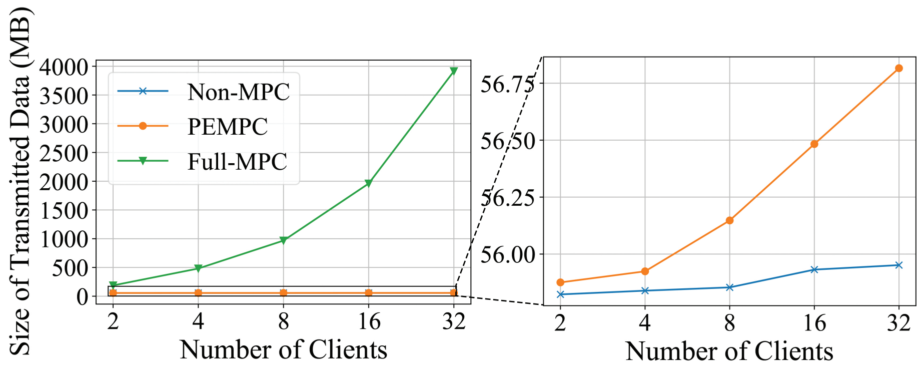
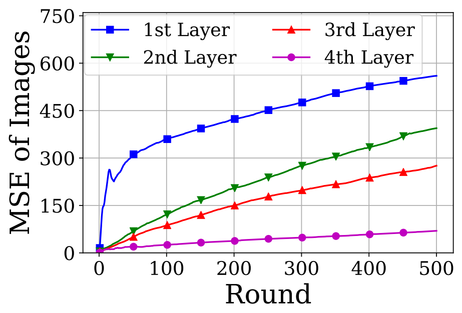


Figure 4. Communication cost on CIFAR-10 dataset with various client numbers.

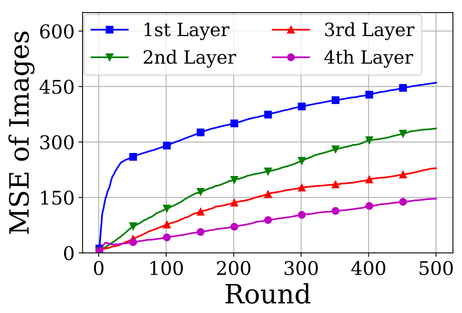
words, the extra communication costs are almost linear to the clients on the network. Considering the facts that client numbers can be large in practice and deep learning model often consists of millions of parameters (*e.g.*, VGG-16) , these extra communication costs are clearly non-trivial. The proposed PEMPC algorithm provides an alternative solution by only encrypting a small proportion of the models and is expected to trigger minimal extra communication cost.

Figure 4 reports the communication costs on CIFAR-10 datasets with various client numbers. On the left figure, we observe that the communication requirements for the full MPC algorithm are significantly higher than the Non-MPC baseline, especially for systems with many clients. In contrast, the curve of PEMPC almost coincides with the Non-MPC, and only by zooming these two curves, we can observe the additional costs triggered by peer-to-peer model transmissions. Similar results can be obtained for the MNIST dataset, hence omitted in this paper.

Along with the communication cost drops is the reduction of local MPC computation time. Since we are only required to perform a local MPC for the first layer, the computational time (CPU: Intel Xeon 5115, GPU: Nvidia 2080Ti) is reduced from 2.7081s for secret shares generation and 0.6951s for secret shares aggregation in full MPC to 0.0019s and 0.0008s in PEMPC.

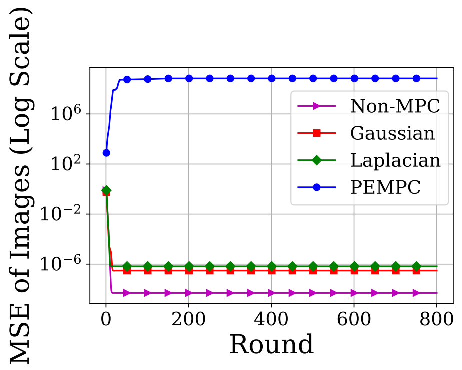


(a) MSE of images on CIFAR10 with DLG.

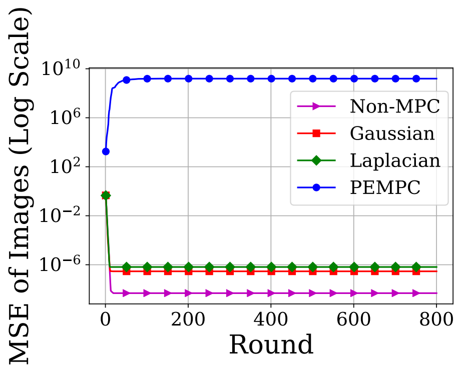


(b) MSE of images on CIFAR-10 with iDLG.

Figure 5. Performance of DLG and iDLG when the gradients of specific hidden layer are encrypted,so the adversaries replaced the encrypted layer with the Gaussian distribution number ($\mu = 0, \sigma = 1$).



(a) MSE of images on MNIST with DLG.



(b) MSE of images on MNIST with iDLG.

Figure 6. Performance of DLG and iDLG on MNIST when the gradients of first hidden layer are replaced with three methods (Gaussian distribution, Laplace distribution, and PEMPC).

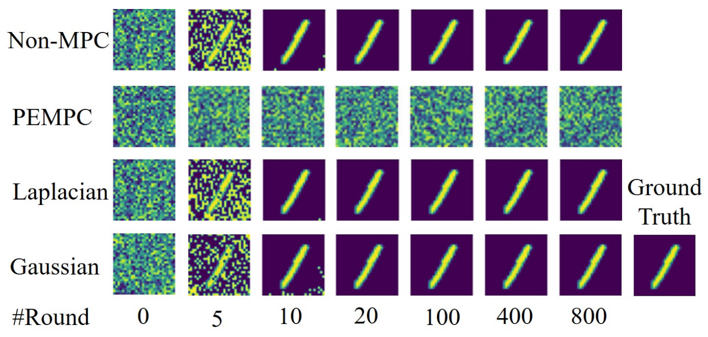
4.3. Performance of DLG and iDLG By Encrypting Partial Weights

In the previous experiments, we propose a partial MPC solution by encrypting only the first layer parameters and obtain significant communication and computation reductions when compared with the full MPC algorithm. Yet, two fundamental questions still await our answers: 1) why the first hidden layer is chosen; 2) whether hiding the first layer of the model is sufficient to prevent the DLG and iDLG attacks.

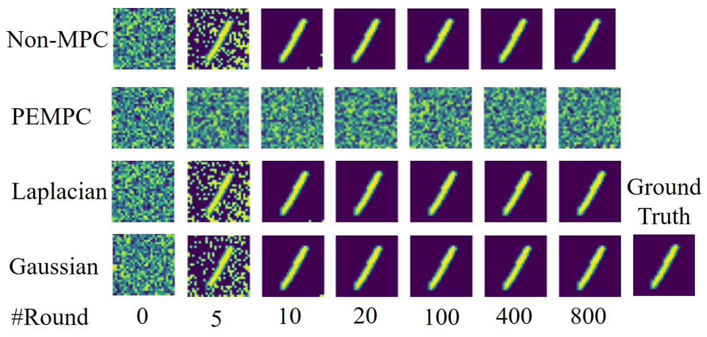
A generic answer to the first question is that the first layer is closest to the original image, but empirical validations may require us to explore the mean-square-error (MSE) of images during the reconstruction process of DLG and iDLG. For better comparison, we reproduce the previous DLG and iDLG methods with their released source codes and use the same architectures as in the original paper. For the MNIST dataset, we replace the layer weights with Gaussian random numbers and test which layer can trigger the largest image MSE, or in other words, hiding parameters in that layer can lead to the best performance. Figure 5(a) and 5(b) report the final performance with both DLG and iDLG, and the results consistently show that encrypting the first layer triggers the largest error. Moreover, the error keeps increasing with the number of iterations.

With this, we can now proceed to the second problem and test whether partial encrypted MPC is sufficient to defend attacks from DLG and iDLG. We calculate the public gradients of the network on a single image from the MNIST dataset. Since our previous observation in Figure 5 suggests that the gradients of the first layer should be hidden, we replace the gradients of the first hidden layer (weight and bias terms) with three methods to see the behavior of DLG and iDLG: 1) Gaussian distribution ($\mu = 0, \sigma = 1$) noise; 2) Laplace distribution ($\mu = 0, \sigma = 1$) noise; 3) the secret shares that generated from the additive secret sharing MPC protocol. After we complete the hidden process, DLG, and iDLG use those gradients to recover the reconstructed image that creates the public shared gradients.

As can be seen in Figure 6, if the malicious server receives the gradients of all hidden layers as a plain text, the reconstruction process is able to obtain a most zero gradient loss and MSE of images (pink line in Figure 6) and reconstructs the training data accurately (Non-MPC row in Figure 7). Yet, when the gradients of the first layer is protected with the proposed method PEMPC, information leakage can be effectively prevented. This can be observed



(a) Results of DLG when the first hidden layer is not hidden and hidden by three methods on MNIST.



(b) Results of iDLG when the first hidden layer is not hidden and hidden by three methods on MNIST.

Figure 7. Reconstructed image from DLG and iDLG on MNIST when the gradients of first hidden layer are replaced with three methods (Gaussian distribution, Laplace distribution, and PEMPC).

in Figure 6, in which the gradient loss and MSE of images (blue, green, and red line do not converge to zero. Similarly, when the gradient loss and MSE of images do not converge to zero, the DLG and iDLG algorithms cannot reconstruct the private images correctly as shonw in the EPMPC row of Figure 7. We also compare PEMPC with the two other baselines that adding the Laplacian and Gausian noises on the original gradients with the magnitude of 10^{-4}. From the results in Figure 6, we can see that both DLG and iDLG can converge and reconstruct the original training images. It is also illustrated in Figure 7, from which we can see that the DLG and iDLG can recovery the training data within 20 rounds.

Based on the above experimental results we verify that PEMPC is an effective way to prevent the DLG and iDLG attacks from reconstructing the original data, and encrypting only the first layer is generally sufficient for against the DLG and iDLG attacks.

4.4. Summary

The experimental results indicate that: 1) the PEMPC strategy maintains a similar prediction accuracy as the Non-MPC baseline while protecting privacy at the same time; and 2) PEMPC requires significantly lower encryption time, inner-node computation and communication costs when compared to the full MPC. These properties make the partially encrypted MPC strategy a competitive encryption solution when dealing with privacy issues in distributed machine learning and prevent model attacks from DLG and iDLG.

5. Conclusion

In this paper, we proposed a new approach (partially encrypted MPC) to prevent indirect information leakage from gradients in distributed machine learning systems. Our proposed method only encrypt part of the gradients using MPC during the model aggregation process. By analysing the data flow of the model aggregation, we observed that the proposed method can prevent recovering the original data from gradients. The experimental results of two widely-used deep learning models (*i.e.*, LeNet-5 and VGG-16) on the MNIST and CIFAR-10 datasets have demonstrated the effectiveness of the proposed partially encrypted MPC encryption strategy, leading to high prediction accuracy (as Non-MPC counterpart), low communication costs and low local computation requirements. In this work, we only provide a raw theoretical foundation of preventing the recovery of the training data. More theoretical analyses and explorations about PEMPC will be the focus of our future work.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] K. Ashton *et al.*, “That ‘internet of things’ thing,” *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[5] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[6] L. Cheng, Y. Wang, Q. Liu, D. H. Epema, C. Liu, Y. Mao, and J. Murphy, “Network-aware locality scheduling for distributed data operators in data centers,” *IEEE Transactions on Parallel and Distributed Systems*, in press. [Online]. Available: https://ieeexplore.ieee.org/document/9329172

[7] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, “Federated learning for mobile keyboard prediction,” *CoRR*, 2018. [Online]. Available: https://arxiv.org/abs/1811.03604

[8] M. Nasr, R. Shokri, and A. Houmansadr, “Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning,” in *IEEE symposium on security and privacy*, 2019, pp. 739–753.

[9] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” in *IEEE Symposium on Security and Privacy*, 2019, pp. 691–706.

[10] B. Hitaj, G. Ateniese, and F. Perez-Cruz, “Deep models under the gan: information leakage from collaborative deep learning,” in *ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 603–618.

[11] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *Advances in Neural Information Processing Systems*, 2019, pp. 14 774–14 784.

[12] B. Zhao, K. R. Mopuri, and H. Bilen, “iDLG: Improved deep leakage from gradients,” *CoRR*, 2020. [Online]. Available: https://arxiv.org/pdf/2001.02610

[13] A. C.-C. Yao, “How to generate and exchange secrets,” in *Annual Symposium on Foundations of Computer Science*. IEEE, 1986, pp. 162–167.

[14] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan, “Secure multi-party computation: theory, practice and applications,” *Information Sciences*, vol. 476, pp. 357–372, 2019.

[15] R. C. Geyer, T. Klein, and M. Nabi, “Differentially private federated learning: A client level perspective,” *CoRR*, 2017. [Online]. Available: https://arxiv.org/abs/1712.07557

[16] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning differentially private recurrent language models,” *CoRR*, 2017. [Online]. Available: https://arxiv.org/abs/1710.06963

[17] R. L. Rivest, L. Adleman, M. L. Dertouzos *et al.*, “On data banks and privacy homomorphisms,” *Foundations of Secure Computation*, vol. 4, no. 11, pp. 169–180, 1978.

[18] R. Canetti, U. Feige, O. Goldreich, and M. Naor, “Adaptively secure multi-party computation,” in *ACM symposium on Theory of Computing*, 1996, pp. 639–648.

[19] O. Goldreich, “Secure multi-party computation,” *Manuscript. Preliminary version*, vol. 78, 1998.

[20] W. Du and M. J. Atallah, “Secure multi-party computation problems and their applications: a review and open problems,” in *Workshop on New Security Paradigms*, 2001, pp. 13–22.

[21] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter *et al.*, “Secure multiparty computation goes live,” in *International Conference on Financial Cryptography and Data Security*. Springer, 2009, pp. 325–343.

[22] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, “Multiparty computation from somewhat homomorphic encryption,” in *Proceedings of the Annual Cryptology Conference*. Springer, 2012, pp. 643–662.

[23] S. Goldwasser, “Multi party computations: past and present,” in *ACM Symposium on Principles of Distributed Computing*, 1997, pp. 1–6.

[24] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[25] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, H. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, “Towards federated learning at scale: System design,” in *Proceedings of the Conference on Machine Learning and Systems*, 2019.

[26] R. Kanagavelu, Z. Li, J. Samsudin, Y. Yang, F. Yang, R. S. M. Goh, M. Cheah, P. Wiwatphonthana, K. Akkarajitsakul, and S. Wang, “Two-phase multi-party computation enabled privacy-preserving federated learning,” in *IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*, 2020.

[27] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, “Inverting gradients–how easy is it to break privacy in federated learning?” *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2003.14053

[28] M. Rigaki and S. Garcia, “A survey of privacy attacks in machine learning,” *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2007.07646

[29] L. Lyu, H. Yu, J. Zhao, and Q. Yang, *Threats to Federated Learning*. Cham: Springer International Publishing, 2020, pp. 3–16.

[30] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[31] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, 2014. [Online]. Available: https://arxiv.org/abs/1804.09081

[32] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[33] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[34] P. Bofill and M. Zibulevsky, “Underdetermined blind source separation using sparse representations,” *Signal processing*, vol. 81, no. 11, pp. 2353–2362, 2001.

[35] L. Zhen, D. Peng, Z. Yi, Y. Xiang, and P. Chen, “Underdetermined blind source separation using sparse coding,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 12, pp. 3102–3108, 2017.

[36] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” *Technical Report TR-2009, University of Toronto*, 2009.