EAI Endorsed Transactions
on Security and Safety

Research Article EAI.EU

Privacy Preserving Collaborative Machine Learning

Zheyuan Liu, Rui Zhang*

University of Delaware, Newark, DE 19716

Abstract

Collaborative machine learning is a promising paradigm that allows multiple participants to jointly train a machine learning model without exposing their private datasets to other parties. Although collaborative machine learning is more privacy-friendly compared with conventional machine learning methods, the intermediate model parameters exchanged among different participants in the training process may still reveal sensitive information about participants' local datasets. In this paper, we introduce a novel privacy-preserving collaborative machine learning mechanism by utilizing two non-colluding servers to perform secure aggregation of the intermediate parameters from participants. Compared with other existing solutions, our solution can achieve the same level of accuracy while incurring significantly lower computational cost.

Received on 23 February 2021; accepted on 15 June 2021; published on 14 July 2021

Keywords: Collaborative Machine Learning, Privacy Preservation, ADMM, Secure Aggregation, Security

doi:10.4108/eai.14-7-2021.170295

1. Introduction

Collaborative machine learning is a promising paradigm for training models from datasets hosted by distributed parties. In contrast to conventional centralized machine learning in which a central server with access to all the training data trains a machine learning model locally, collaborative machine learning allows multiple parties each with a local dataset to jointly train a global model over the whole dataset without revealing any party's local dataset to others. Collaborative machine learning is particularly attractive when local datasets involve highly sensitive information such as health records.

Unfortunately, even though the local training dataset of each participant is kept secret from other parties during the training process, intermediate model parameters exchanged among different participants during the training process may still reveal some information about the local dataset, which may be used to infer or even recover the local dataset of a target participant [1–4]. In particular, recent studies have shown that it is possible to reconstruct local input data from gradient information in collaborative learning [5]. Therefore, it is important to design a sound mechanism to prevent such privacy leakage in collaborative machine learning.

Existing solutions for protecting participants' data privacy in collaborative machine learning can be broadly divided into two categories. The first category [6–11] uses cryptographic techniques to encrypt the intermediate model parameters while still allowing global model updates. Although these solutions can ensure the confidentiality of local model parameters during the training process, encryption and decryption operations usually incur high computation and communication cost, which may be even infeasible for resource-constrained mobile devices. The second category [12–17] adopt the Differential Privacy (DP) paradigm to have each party randomly perturbs its intermediate model parameters to prevent others from inferring its local dataset while still allowing a reasonably accurate model to be trained. In comparison with the encryption-based solutions, DP-based collaborative machine learning methods are easier to deploy and incur much lower computation cost. However, there is an inherent trade-off between the level of privacy guarantee and the accuracy of the trained model. In addition, the random noise introduced in every iteration of the training process may make the training process converge much slower.

In this paper, we tackle this challenge by introducing a novel privacy-preserving collaborative machine

*Corresponding author. Email: ruizhang@udel.edu

learning mechanism. Our mechanism explores two non-colluding servers and efficient cryptographic techniques to realize secure aggregation of local model parameters during the training process and protect the privacy of the local dataset of each participant. Compared with existing solutions, our solution can train a global model with the same accuracy of standard machine learning methods while incurring very low computation and communication overheads. Our contributions in this paper can be summarized as follows.

- We a novel privacy-preserving collaborative machine learning mechanism that explores two non-colluding servers and efficient cryptographic primitives.
- Our mechanism allows multiple parties to train an accurate global model without revealing any party's local dataset to each other.
- We confirm the efficacy and efficiency of the proposed mechanism via detailed simulation studies.

The rest of this paper is structured as follows. Section 2 discusses the related work. Section 3 formulates the problem to be solved and presents the system model, adversary model, and our proposed privacy preserving mechanism. Section 5 describes the experiments we conduct to evaluate the performance of our proposed mechanism. Section 6 finally concludes this paper.

2. Related Work

Various solutions have been proposed to protect the data privacy of participants in collaborative machine learning, which can be broadly classified into two categories: encryption-based solutions and Differential Privacy (DP)-based solutions.

Encryption-based solutions protect the intermediate model updates through Secure Multi-Party Computation (SMPC). SMPC was developed for the scenarios where multiple parties wish to jointly evaluate a function over their private data without revealing any party's data to other parties. Different SMPC techniques have been used to realize privacy-preserving collaborative learning, including Yao's garbled circuit protocol [6, 7], homomorphic encryption [8, 9], secure aggregation methods [10, 11], etc. These encryption based mechanisms can produce accurate model without harming the prediction accuracy of the trained model because they do not change the model parameters. On the other hand, since these techniques commonly involve expensive public key operations, they usually incur high computation and communication costs.

DP-based solutions protect participants' data privacy by having each participant randomly perturb its local model parameters. Different DP-based solutions add random noises to different model parameters, including local model parameters [12–14], local objective functionss [15, 16], and local training datasets [17]. DP-based collaborative machine learning methods provide a tunable balance between data privacy and model utility. The additional computation cost introduced by the perturbation is also small, making DP-based solutions more efficient than encryption-based methods. However, the noises introduced during the training process result in the decrease in the accuracy of the trained model.

3. Problem Formulation

In this section, we introduce the system and adversary models along with our design goals.

3.1. System Model

We consider a system in which a central server and N participants collaboratively trains a global model. We use the Alternating Direction Method of Multipliers (ADMM) [18] as our machine learning method, which is an promising machine learning framework and has been attract a lot of attentions in recent years due to its capability to support a wide range of objective functions and mild constrains on objective functions such as weak convexity.

In ADMM, participants minimize their local loss functions based on their local dataset and reach consensus with others to train a global model. The constrained collaborative optimization problem can be formulated as following:

$$\min_{x_i \in \mathbb{R}^D, i \in 1,...,N} \sum_{i=1}^{N} f_i(x_i) \quad \text{subject to } x_i = z, \forall_i \in 1, ..., N \tag{1}$$

where $x_i \in \mathbb{R}^D$ is participant i's local copy of the model parameter to learn, and f_i is participant i's local loss function. A consensus over $x_1, ..., x_N$ needs to be reached with the global copy of the model z to complete the training process.

Under the ADMM framework, Eq. (1) can be rewritten in the augmented Lagrangian form

$$\mathcal{L}_\rho(x_1, ..., x_N, z, \lambda) = \sum_{i=1}^{N} (f_i(x_i) + \lambda_i(x_i - z) + \frac{\rho}{2}\|x_i - z\|_2^2) \tag{2}$$

where $\lambda_1, ..., \lambda_N$ are the augmented Lagrange multipliers and ρ is the penalty term for regulation. Eq. (2) can

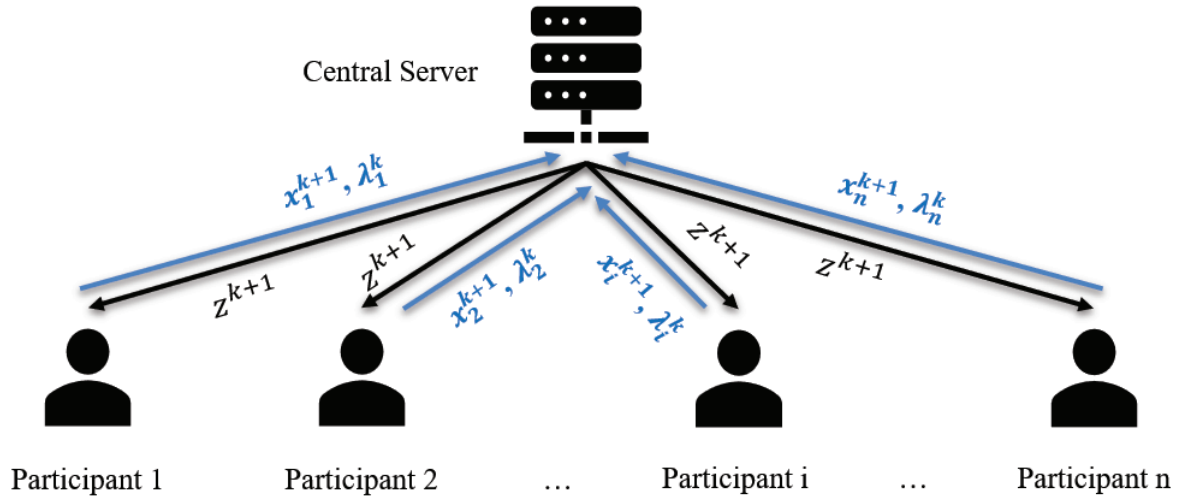


Figure 1. ADMM based collaborative machine learning model

be further transformed into alternative x-update and z-update in ADMM as follows.

$$\begin{aligned} x_i^{k+1} &= \underset{x_i}{\operatorname{argmin}}(f_i(x_i) + \lambda_i^k(x_i - z^k) + \frac{\rho}{2}\|x_i - z^k\|_2^2) \\ z^{k+1} &= \frac{1}{N}\sum_{i=1}^{N}(x_i^{k+1} + \frac{1}{\rho}\lambda_i^k) \\ \lambda_i^{k+1} &= \lambda_i^k + \rho(x_i^{k+1} - z^{k+1}) \end{aligned} \tag{3}$$

x_i^{k+1}, z^{k+1}, λ_i^{k+1} are values updated in the $(k+1)^{th}$ iteration.

The high-level system structure and the training process is illustrated as Fig. 1. Participants are responsible to update and maintain their own x values, while the server aggregates the local updated x values from all participants and updates the global z value. The system works in a synchronous fashion. In the $(k+1)$th iteration, each participant i uses the current global model parameter z^k to calculate the new x value x_i^{k+1} according to the x-update step in Eq. (3) and then sends x_i^{k+1} and λ_i^k to the server. The central server gathers the received x values and λ values from the N participants and calculate new global parameter z^{k+1} based on the z-update step in Eq. (3) and send it to all participants. After receiving the new global parameter z^{k+1}, each participant i calculates the new λ value based on the λ-update step in Eq. (3). This alternative and iterative parameter updating process terminates when the change in z and the maximum difference between x_i and z between two adjacent iterations are both smaller than predefined thresholds.

In this collaborative learning mechanism, the server only requires participants to update their x and λ values to train the final model, thus participants can keep their local dataset unexposed to other parties.

3.2. Adversary Model

We assume that N participants and a central server jointly train a machine learning model using the ADMM-based collaborative learning algorithm. We assume that the adversary is either be an honest-but-curious central server or a malicious participant that engages in the collaborative training and eavesdrops the updated parameters from the victim. The adversary's goal is to infer a victim participant's private dataset via inference attacks.

We assume that the adversary can observe the communication between the server and the victim participant. Specifically, the adversary knows the z values sent by the server, and the x and λ values sent by the victim participant in each iteration. However, we assume that the adversary does not have the direct access to the victim's local dataset.

We stress that if the accurate global model is published by the end of training, the information leakage of a participant's local dataset caused by the final model itself is inevitable. Therefore, we only seek to prevent the information leakage caused by the intermediate parameters in the training process alone.

4. The Proposed Approach

As discussed in Section 2, existing privacy preserving collaborative learning solutions have different limitations. DP-based solutions produces an inaccurate final model, while encryption-based methods incur high computation cost. To tackle with these limitations, we seek to design a collaborative machine learning mechanism that simultaneously achieves high model accuracy, strong privacy guarantee for local participants, and high computation efficiency.

Inspired by the secure aggregation mechanisms [10, 11], we observe that the iterative updates of the global model parameter can be considered as a SUM aggregation of local model parameters. Therefore, we propose a light-weight privacy preserving collaborative learning method that explores two non-colluding servers to allow participants to train an accurate model efficiently without revealing information of their private datasets. Such two non-colluding servers have also been explored in other secure multiparty computation applications such as [19].

We assume there are two non-colluding servers, including a primary server S_1 and an auxiliary server S_2. Server S_1 is responsible for the global model updates, and server S_2 is merely used for privacy provision. We assume that each participant i shares a secret key K_i with the S_2 via some secure channels. The proposed privacy preserving collaborative machine learning algorithm works as follows.

Before the first iteration, server S_1 informs the IDs of all the participants to S_2 and then randomly selects its initial z^0 value and broadcasts it to all the N participants. Each participant i selects its initial λ_i^0. In the $(k+1)$th iteration ($k \geq 0$), each participant i first executes the x-step according to Eq. (3) as

$$x_i^{k+1} = \underset{x_i}{\operatorname{argmin}}(f_i(x_i) + \lambda_i^k(x_i - z^k) + \frac{\rho}{2}\|x_i - z^k\|_2^2) \quad (4)$$

and computes

$$\lambda_i^{k+1} = \lambda_i^k + \rho(x_i^{k+1} - z^k)\,. \quad (5)$$

Let $H : \{0,1\}^* \rightarrow \mathbb{R}^D$ be a cryptographic hash function that maps any input into a D-dimensional vector. In particular, suppose that each λ_i^k and x_i^k is represented as a q-bit binary number, i.e., $\lambda_i^k, x_i^k \in \{0, \ldots, 2^q - 1\}$. Given a cryptographic hash function $H'(\cdot)$ that maps any input to a digest of h bits, where $h \geq qD$, we can realize $H(\cdot)$ by defining

$$H(x) = H'(x) \mod 2^{qD}\,.$$

The participant i then generates a random noise $r_i^{k+1} = H(K_i\|k)$, where K_i is the secret key shared with S_2. He then computes

$$y_i^{k+1} = x_i^{k+1} + \frac{1}{\rho}\lambda_i^k + r_i^{k+1} \quad (6)$$

and send y_i^{k+1} to S_1.

After receiving $y_1^{k+1}, \ldots, y_N^{k+1}$, server S_1 informs S_2 that it has received the local model parameters from all participants. S_2 then computes

$$r^{k+1} = \sum_{i=1}^{N} H(K_i\|k) \quad (7)$$

and sends r^{k+1} to S_1.

On receiving r^{k+1}, server S_1 then computes

$$\begin{aligned} z^{k+1} &= \frac{1}{N}\sum_{i=1}^{N} y_i^{k+1} - r^{k+1} \\ &= \frac{1}{N}(\sum_{i=1}^{N}(x_i^{k+1} + \frac{1}{\rho}\lambda_i^k + r_i^{k+1})) - r^{k+1} \\ &= \frac{1}{N}(\sum_{i=1}^{N}(x_i^{k+1} + \frac{1}{\rho}\lambda_i^k + H(K_i\|k))) - \sum_{i=1}^{N} H(K_i\|k) \\ &= \frac{1}{N}\sum_{i=1}^{N}(x_i^{k+1} + \frac{1}{\rho}\lambda_i^k) \end{aligned} \quad (8)$$

and send it to all participants.

After receiving the new global parameter z^{k+1}, each participant i calculates the new λ value based on the λ-update step in Eq. (3), and the $(k+1)^{th}$ iteration ends. Fig. 2 illustrates the proposed privacy preserving collaborative learning system.

As we can see from Eq. (7) and Eq. (8), during the training process, server S_1 does not learn any of the exact local parameters $x_1^{k+1}, \ldots, x_N^{k+1}$ and $\lambda_1^{k+1}, \ldots, \lambda_N^{k+1}$ from any individual participant. Since S_1 does not know the secret key shared between each participant and S_2, it cannot remove r_i^{k+1} from y_i^{k+1} it receives to recover x_i^{k+1} or λ_i^{k+1} for any participant i. Meanwhile, what S_2 has at hand are only secret keys shared with participants, and the current iteration number that S_1 announces, it is impossible for S_2 to infer the values of $x_1^{k+1}, \ldots, x_N^{k+1}$ and $\lambda_1^{k+1}, \ldots, \lambda_N^{k+1}$ either. Therefore, as long as S_1 and S_2 do not collude with each other, neither of them can learn the values of $x_1^{k+1}, \ldots, x_N^{k+1}$ and $\lambda_1^{k+1}, \ldots, \lambda_N^{k+1}$ in each iteration. The data privacy of individual participants is thus preserved.

We can also see from Eq. 8 that S_1 can still compute an accurate global model parameter z^k in each iteration, so the utility of the final model is not harmed. In contrast to existing encryption-based solutions that involve expensive public key operations, the cryptographic hash function that we use incurs a much lower computation cost. There is only one extra simple computation required from both S_1 and each participant, compared with the non-privacy-preserving version. There is also only one more round of message exchange between S_1 and S_2 in each iteration. Therefore, our design goal has been fulfilled: high accuracy of the model, high privacy preservation level of participant's data, and low computation cost.

5. Performance Evaluation

In this section, we report our experiment results.

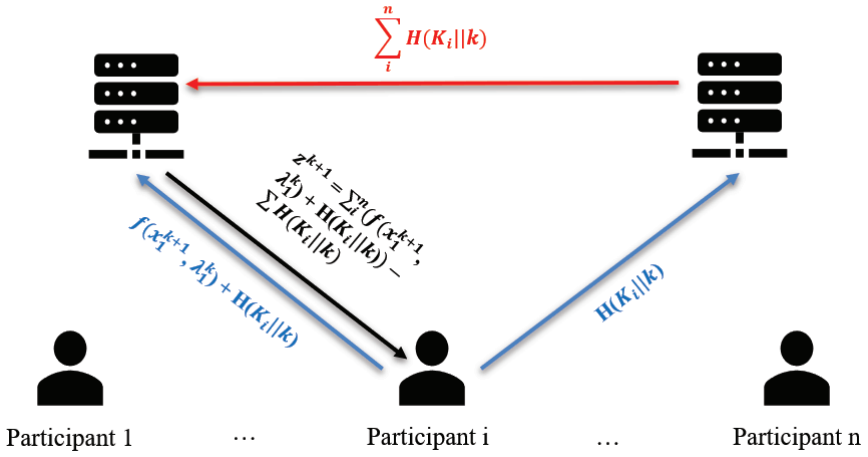


Figure 2. Privacy-preserving collaborative machine learning

5.1. Experiment Setup

We implemented our Privacy Preservation ADMM-based Collaborative Learning algorithm (PPADMM) based on Least Absolute Shrinkage and Selection Operator (Lasso) as it can be easily extended to a wide variety of statistical model. In addition, we choose SHA-256 as the cryptographic hash function in our algorithm. We implemented our solution using Python 3. Our training and testing dataset in this experiment is generated by the in-built regression distribution generator in *sklearn.datasets*, a library developed specifically for machine-learning purpose in Python 3. *sklearn.datasets* can generate a dataset that satisfies a certain Gaussian distribution with tunable mean values and noise. The software is run on a laptop with a 2.60 GHZ CPU that has 8 Intel i7-6700HQ cores and 16GB RAM.

5.2. Experiment Design

Collaborative Learning System. The collaborative machine learning system consists of 10 participants and 2 non-colluding servers S_1 and S_2. S_1 serves as the primary server which gathers local parameters and updates the global model, and S_2 is responsible for privacy provision. S_1, S_2 and participants update global model and local models iteratively following the protocol described in Section 4.

Dataset and System Parameters. We generate 22000 synthesized samples following a Gaussian distribution, each with 10 features, and use them to train and test our Lasso predictor. The size of the training set is 20000, and the size of the test set is 2000. The standard deviation of the gaussian noise of the dataset is 20. We set the learning rate α to be 0.001, and the regulation parameter ρ to avoid overfitting to be 1, which are the common parameter settings for Lasso regression.

Performance Metrics. We evaluate the performance of our algorithm with following metrics.

Accuracy: We use 2 metrics to measure the accuracy of our algorithm. The first one is the Root Mean Square Error (RMSE) defined as follows.

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(\hat{y_i} - y_i)^2}{N}} \tag{9}$$

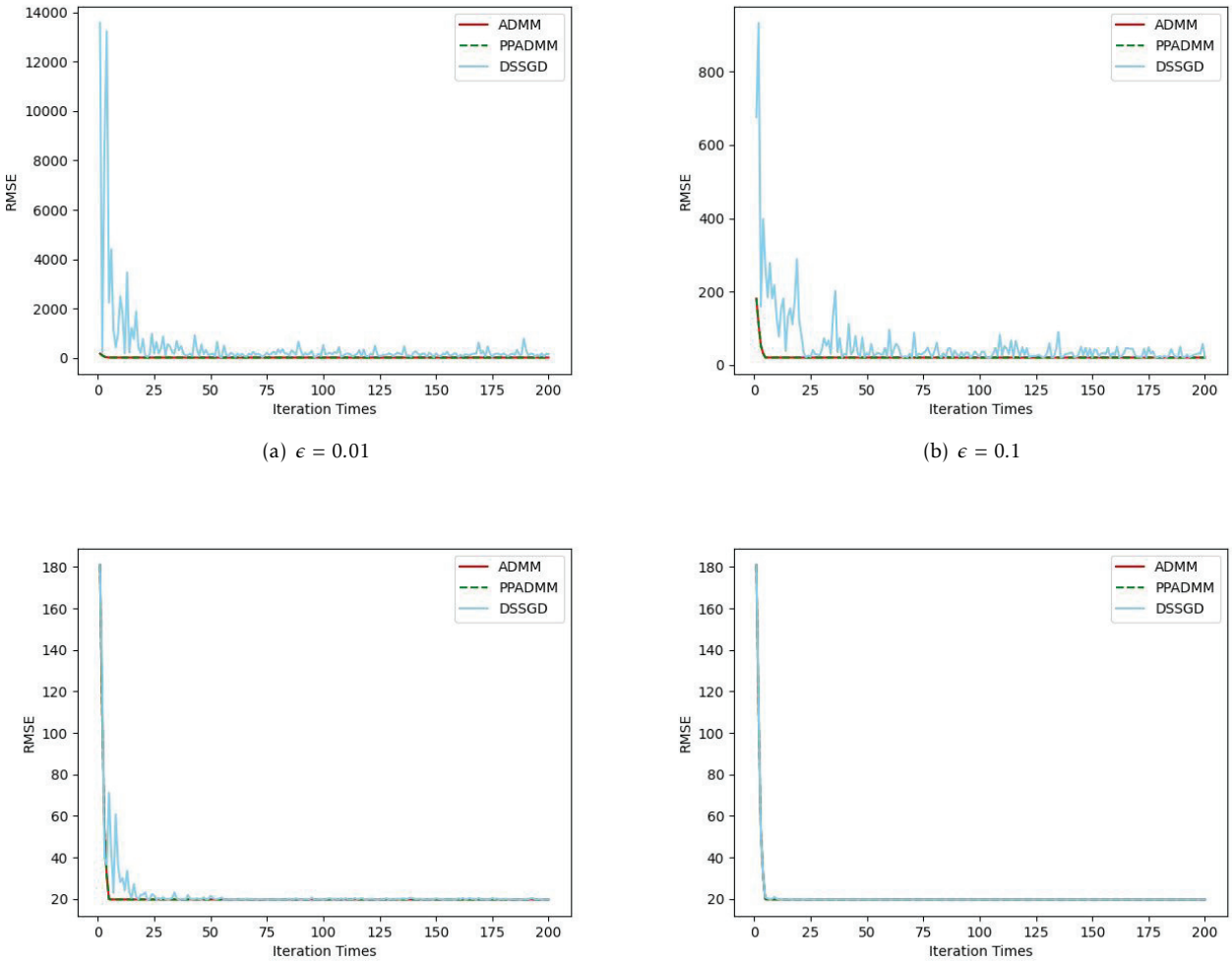
where y_i is the true label of sample i, $\hat{y_i}$ is the label predicted by the model, and N is the total number of samples. RMSE is a common method used to evaluate the prediction error of a regression-based model. It reflects the differences between the true values and the predicted values of samples. The lower the RMSE is, the more accurate the model is.

The second metric is the Model Parameter Euclidean Distance (MPED), which is defined as follows.

$$MPED = \left\|W - \hat{W}\right\|_2 = \sqrt{\sum_{i=1}^{m} (w_i - \hat{w_i})^2} \tag{10}$$

where W is the true regression function that generates the training dataset, and $\hat{W}$ is the predictor trained with our algorithm. w_i and $\hat{w_i}$ are the i^{th} parameter of W and $\hat{W}$ respectively, and m is the total number of parameters. MPED represents the difference between the parameters of the true regression function and the predicted regression function. Smaller MPED indicates that the predicted regression is closer to the true distribution of data in the dataset.

Computation Cost: We measure the computation cost of our algorithm when it is run with a certain



(a) $\epsilon = 0.01$ (b) $\epsilon = 0.1$

(c) $\epsilon = 1$ (d) $\epsilon = 10$

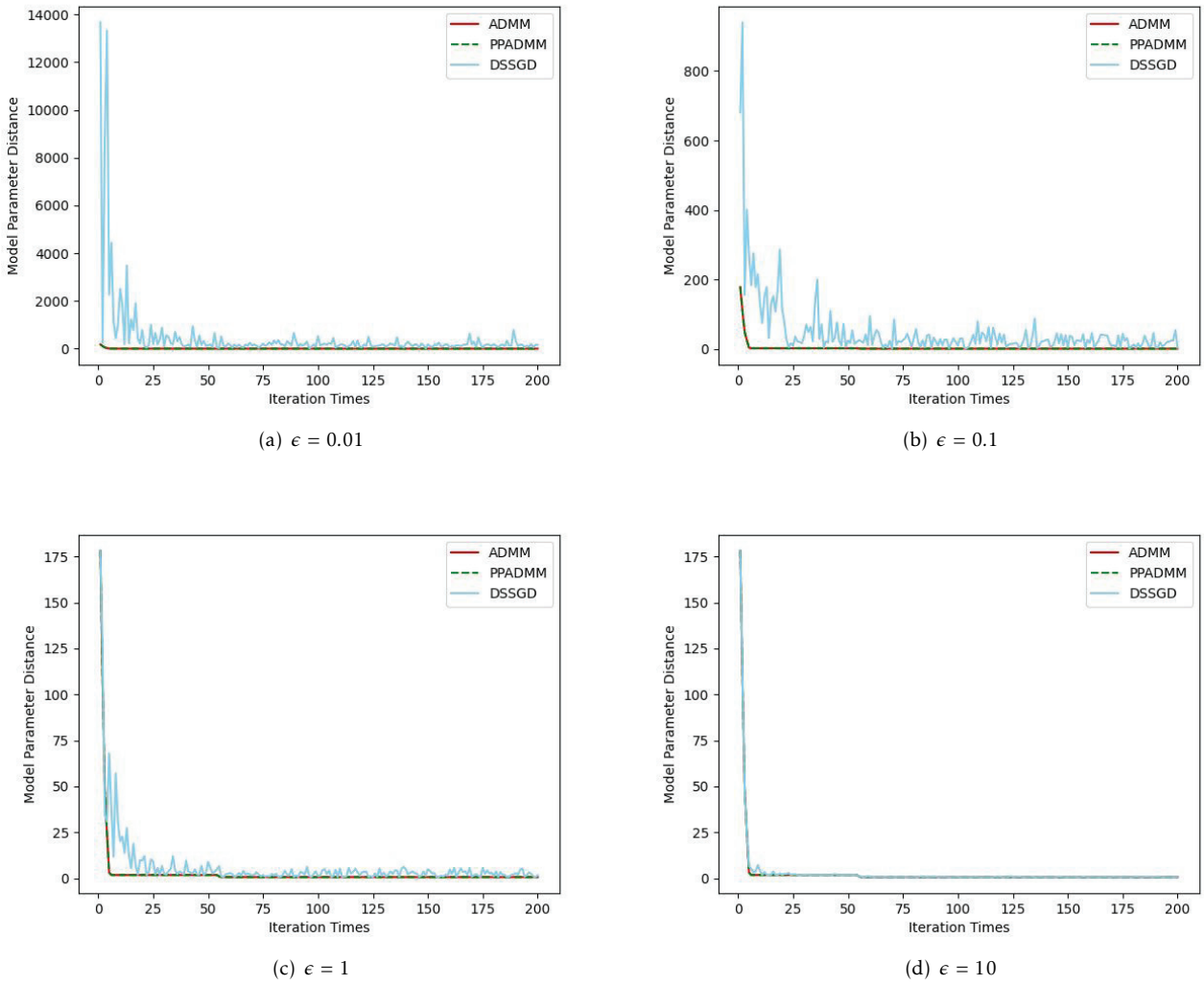
Figure 3. Relationship between RMSE and the number of iteration of ADMM, PPADMM, and DSSGD.

iteration number. We consider the actual running time of the algorithm as the computation cost in this experiment.

5.3. Experimental Results

We compare three algorithms: the original ADMM-Lasso, our algorithm PPADMM, and a classic DP-based privacy preserving collaborative learning algorithm, DSSGD [12], with respect to their performance on model accuracy and computational efficiency using the metrics in Section 5.2.

We measure both RMSE and MPED to evaluate the accuracy of the regression models trained by all these three algorithms with different privacy parameters $\epsilon = 0.01, 0.1, 1, 10$, respectively. The result is shown in Fig. 3. We can see that our algorithm PPADMM has the same RMSE as the original ADMM-Lasso algorithm. This is because during the secure aggregation process of PPADMM, the primary server S_1 removes the aggregated noise with the help of the auxiliary server S_2 to obtain the exact original global model parameters, thus the final model produced by PPADMM and ADMM-Lasso are identical. Since in DSSGD the participants perturbs the local parameters with Laplace noise in every iteration, it results in a model that predicts labels of data with a much higher RMSE compared with PPADMM and ADMM-Lasso. For example, RMSE of DSSGD can still be larger than 100 when $\epsilon = 0.1$ after it converges, while PPADMM and ADMM-Lasso has a RMSE close to 20, which is the standard deviation of the gaussian noise of the dataset. The convergence speed of PPADMM and ADMM-Lasso is also faster than that of DSSGD, according to Fig. 3. We can also see that there is an inherent trade-off between



(a) $\epsilon = 0.01$

(b) $\epsilon = 0.1$

(c) $\epsilon = 1$

(d) $\epsilon = 10$

Figure 4. Relationship between MPED and iteration number under ADMM, PPADMM, and DSSGD

the data privacy and model utility in a DP-based mechanism. The RMSE of predicted labels generated by DSSGD decreases as the privacy parameter ϵ increases, indicating that the accuracy of the model increases as the privacy guarantee decreases. Such trade-off does not appear in PPADMM and ADMM-Lasso, as RMSE of labels predicted in both mechanisms are independent of the privacy parameter ϵ.

We also measure how Model Parameter Euclidean Distance (MPED) of three algorithms (see Eq. (10)) change with the iteration number as the other criteria for model accuracy. The result is shown in Fig. 4. We can find that similar to the measurement of RMSE, PPADMM and ADMM-Lasso generate smaller MPED under different privacy parameters ϵ compared to DSSGD, which means our algorithm trains a model closer to the true distribution of original dataset.

Fig. 5 shows the computation time of three algorithms. The x-axis represents the iteration number that the algorithm runs, and the y-axis represents the real-world computation time of the algorithm spends with such number of iterations. We can see that PPADMM has a higher computation cost compared with ADMM-Lasso and DSSGD. The main reason that PPADMM incurs higher computational cost is that even though we apply SHA-256 as our cryptographic hash function, and it is more efficient than cryptographic techniques such as Yao's Garbled Circuit and homomorphic encryption used in other existing cryptographic-based privacy preserving collaborative algorithms, SHA-256 itself is still more time-consuming compared with the generation of Laplace noise in DSSGD. Although our algorithm has a higher computation cost than DSSGD, we can see that the difference between these two is not

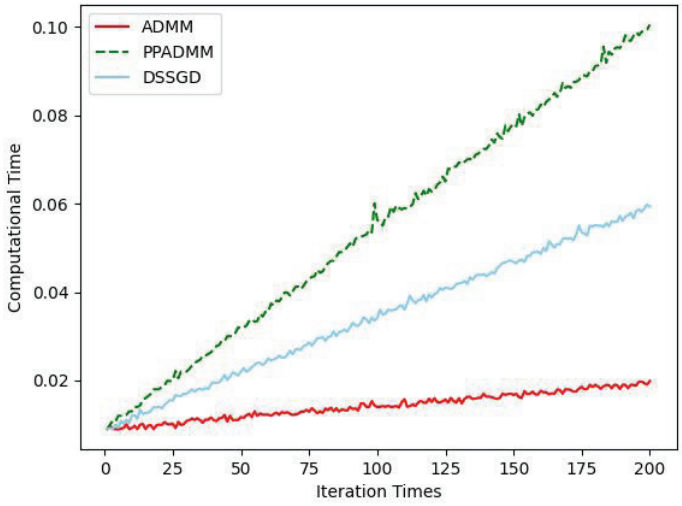


Figure 5. Comparison of the computation time of ADMM, PPADMM, and DSSGD

unacceptably large, and unlike other cryptographic-based mechanisms [6, 8, 11], our algorithm results in a linear-growth computational cost. We let participants and S_2 apply SHA-256 to generate secure hash digests at each iteration in our implementation, allowing S_2 to change the secret key shared with a participant during the training process. An more computationally efficient approach for participants and S_2 is that they can pre-compute the hash digest before the collaborative learning starts, if they agree on not changing the shared secret key in the middle of the training.

Since the information leakage of participants' local datasets caused by exchanged intermediates in collaborative learning is fairly complex, and it varies corresponding to different types of inference attacks that the adversary launches, there is yet no universal criterion that quantitatively measures such privacy leakage in a collaborative learning system. We argue that our algorithm provides stronger privacy guarantee than DP-based mechanisms, since the intermediate local parameters exchanged in PPADMM is merely the cryptographic hash digest of the original ones. It is computationally infeasible for the adversary to acquire the original local parameters if he does not know the secret key shared between participants and S_2, due to the one-way property and collision-free property of the cryptographic hash function. On the other hand, DP-based mechanisms provide privacy protection by adding noise generated from a certain distribution such as Laplace distribution. Such distribution can be estimated through multiple rounds of observation on the perturbed intermediates, and such estimation could be used to reduce the impact of perturbation on these intermediates and infer the distribution of original intermediates.

6. Conclusions and Future Work

In this paper, we introduce a novel privacy-preserving collaborative learning mechanism based on secure SUM aggregation via two non-colluding servers. Our solution allows the server to receive accurate aggregated local model update in each iteration without learning any individual participant's local model update and can achieve the same level of accuracy of standard collaborative learning mechanisms. Built upon efficient cryptographic primitives, the computation cost of our mechanism is also orders of magnitude lower than existing encryption-based solution. We have confirmed the efficacy and efficacy of our mechanism through experiment studies.

References

[1] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[2] B. Hitaj, G. Ateniese, and 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.

[3] 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 *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

[4] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative

learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 691–706.

[5] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[6] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Secure linear regression on vertically partitioned datasets." *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 892, 2016.

[7] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 19–38.

[8] C. Zhang, M. Ahmad, and Y. Wang, "Admm based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 565–580, 2018.

[9] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 334–348.

[10] G. Danner and M. Jelasity, "Fully distributed privacy preserving mini-batch gradient descent learning," in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2015, pp. 30–44.

[11] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[12] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[13] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan, "Crowd-ml: A privacy-preserving learning framework for a crowd of smart devices," in *2015 IEEE 35th International Conference on Distributed Computing Systems*. IEEE, 2015, pp. 11–20.

[14] T. Zhang and Q. Zhu, "Dynamic differential privacy for admm-based distributed classification learning," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 172–187, 2016.

[15] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via functional perturbation," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 395–408, 2016.

[16] Y. Wang, M. Hale, M. Egerstedt, and G. E. Dullerud, "Differentially private objective functions in distributed cloud-based optimization," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 3688–3694.

[17] B. Cyphers and K. Veeramachaneni, "Anonml: Locally private machine learning over a network of peers," in *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2017, pp. 549–560.

[18] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[19] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *2017 IEEE 33rd international conference on data engineering (ICDE)*. IEEE, 2017, pp. 633–644.