Augmented Multi-Party Computation Against Gradient Leakage in Federated Learning

Chi Zhang, Sotthiwat Ekanut, Liangli Zhen, and Zengxiang Li

Abstract—Multi-Party Computation (MPC) provides an effective cryptographic solution for distributed computing systems so that local models with sensitive information are encrypted before sending to the centralized servers for aggregation. Though direct local knowledge leakages are eliminated in MPC-based algorithms, we observe the server can still obtain the local information indirectly in many scenarios, or even reveal the groundtruth images through methods like Deep Leakage from Gradients (DLG). To eliminate such possibilities and provide stronger protections, we propose an augmented MPC approach by encrypting local models with two rounds of decomposition before transmitting to the server. The proposed solution allows us to remove the constraint that servers must be honest in the general federated learning settings since the true global model is hidden from the servers. Specifically, the augmented MPC algorithm encodes local models into multiple secret shares in the first round, then each share is furthermore split into a public share and a private share. Consequences of such a two-round decomposition are that the augmented algorithm fully inherits the advantages of standard MPC by providing lossless encryption and decryption while simultaneously rendering the global model invisible to the central server. Both theoretical analysis and experimental verification demonstrate that such an augmented solution can provide stronger protections for the security and privacy of the training data, with minimal extra communication and computation costs incurred.

Index Terms—Privacy-Preserving, Multi-Party Computation, Federated Learning, Data Leakage

1 Introduction

Data in many scenarios [1]–[3] are naturally distributed or owned by different organizations/users, and due to privacy, security, and administrative regulations, there generally exist many limitations on uploading them across countries or institutions for traditional centralized learning. In such cases, researchers often prefer to construct the decision-making models in a distributed learning way [4]–[8] so that multiple participants can build a joint machine learning model without sharing their local data, *i.e.*, no need to upload the private data to the cloud or exchange data across participants. But recent studies show that sharing local models instead of data could still leak sensitive information to other participants or the central server. For instance, Carlini *et al.* [9] demonstrated that some sensitive information, *e.g.,* credit card numbers, can be extracted from a recurrent neural network trained on users' email messages. Zhu *et al.* [10] and Zhao *et al.* [11] further showed that the original data can also be recovered from the local gradients.

For this reason, in federated learning, models of local participants generally will go through certain encryption steps before being transmitted to the server. One typical example of such an encryption procedure would be the differential privacy (DP) method [12]–[14], which binds certain levels of noise to the original data or uses generalization methods to obscure specific sensitive attributes until the third party cannot distinguish the individual. Such a local blurring step possesses strong information-theoretic guarantees and also can be easily implemented, but it also leads to certain accuracy drops for practical applications.

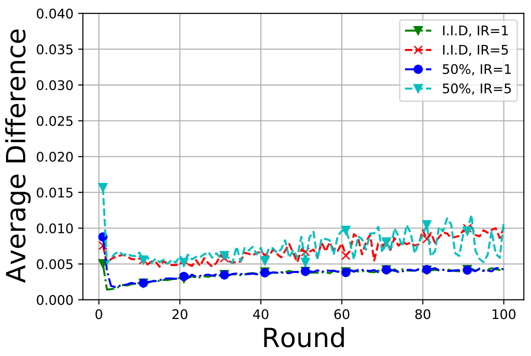


Fig. 1: The average L_1-norm difference between the aggregated global model and the first participant' counterpart on LeNet-5. "50%" refers to keeping 50% images of one class on a node and evenly distributing the remaining images, and IR denotes the inner round. The average model difference tends to be minor in all scenarios, especially with small IR.

Another promising solution for secure learning is Multi-Party Computation (MPC), which can date back to the early studies in [15], [16]. The MPC method, generally considered a cryptographic solution, provides a generic primitive that enables multiple parties to jointly compute an arbitrary functionality without revealing their private inputs [17]–[20]. It addresses the privacy problem of cooperative computation in a secure fashion so that each participant only obtains certain pieces from neighborhood models. Unlike the DP methods, it allows the model encryption and decryption to be performed in a lossless way so that local participants can fully recover the true global model without suffering from accuracy drops.

- *This work was supported by the A*STAR CRF/ATR Grant: A Secure and Privacy Preserving AI Platform for Digital Health.*
- *Chi Zhang, Sotthiwat Ekanut and Liangli Zhen are with the Institute of High Performance Computing, A*STAR, Singapore. E-mail: {zhang_chi, sotthiwat_ekanut_from.tp, zhenll}@ihpc.a-star.edu.sg.*
- *Sotthiwat Ekanut is also with the National University of Singapore.*
- *Zengxiang Li is with the Digital Research Institute, ENN Group, China. E-mail: lizengxiang@enn.cn.*
- *Corresponding Author: Liangli Zhen.*

Though MPC is designed to prevent the local model leakages and adopted in many open-source federated learning libraries (e.g., PySyft), we notice there still exist backdoors where the aggregation center can fully recover or obtain a sufficient-close model to the local counterparts or even reconstruct local images from received gradient information. As such, utilizing the MPC technique directly to federated learning applications may lead to an insecure situation that violates the purpose of MPC itself. To give a simple example, we consider the case where data on local participants are exactly the same; then, the global model will be identical to each local model. In other words, the MPC mechanism indeed fails in this case as the server can recover each local model. Such an exact match is, of course, rather artificial. However, in practical machine learning, we observe that the difference between the global model and local models can be minor after certain learning rounds, as reported in Figure 1. The underlying reason is that the model aggregation step in federated learning often renders local models to be trained from the same starting point and the local updates can be minor in the following training steps, especially with smaller inner rounds or updating local parameters in a conservative way like FedProx [21]. These experimental insights are also validated by theoretical justifications [22], [23] that deep learning models often move towards the intersection of local optimums so that the difference between local models tends to be minor.

Moreover, allowing the central server to obtain the joint model/gradients fully could even lead to data leakage by using recent reverse engineering techniques like DLG [10] and iDLG [11]. When sending local gradients to the server, MPC methods allow each participant to shatter its own local gradients, but after averaging, the server can still receive the unbiased averaged gradients. By considering all the local training images as one batch, DLG would allow the server to recover training images from local participants through gradient matching.

Motivations of our research are not merely amending this potential issue of MPC, though MPC itself tends to be vital for complete privacy-preserving on knowledge-based systems. Note that a typical assumption for federated learning is that the local participants are honest, whereas the server is honest-but-curious [7]. While the former assumption is generally easy to satisfy since all participants are stakeholders, the requirement of ensuring the third-party aggregator to be honest often tends out to be non-trivial in practice. Public aggregation platforms like Amazon Web Services (AWS) generally claim that they do not store or use the global model, but privacy-sensitive users like banks may still prefer these platforms not to obtain the global model. Moreover, for fairness considerations [24], these platforms contribute only an aggregation step and should not take the free-ride to obtain the true global model.

For these two reasons, it is necessary to design a new federated learning mechanism so that the aggregator is denied access to any local information, even the true global model itself, but in the meanwhile still functions as a normal fusion center. The solution we provide in this paper, named Augmented Multi-Party Computation (AMPC), utilizes the inter-participant communications of standard MPC and encrypts each local model with two rounds of decomposition before communicating with the aggregation center. Specifically, the local models are encoded into multiple secret shares in the first round. Then each share is split into two parts in the second round: one public share to be sent to the server for model aggregation and one private share stored on the local machine to recover the true global model. Theoretical analysis and practical experiments demonstrate the advantages of such a new paradigm in the following aspects:

1) *More secure scheme*: As an augmented approach, the proposed algorithm addresses the information leakage issues and maximally eliminates the possibility of obtaining the local models on the server side. It only grants the server access to some randomly biased public models, which can only predict like random guesses in real-world experiments.
2) *Inherit merits of MPC*: Standing out as a more secure aggregation method, the AMPC algorithm also inherits the merits of standard MPC where the local participant can fully recover the underlying true global model and the additional encryption and decryption steps do not lead to accuracy drops.
3) *Weaker assumption on server*: Local participants only need to transmit part of their models to the server and hold private secret shares on local devices. As a consequence, there is no necessity to assume that the third party is not honest; thus, finding such an aggregator tends to be simple in practice.

2 The Proposed Method

2.1 Preliminary

Federated learning enables multiple participants to collaboratively learn a shared decision-making model while keeping all the training data $\mathcal{X} = \{X_1, X_2, \ldots, X_m\}$ on m local participants. A coordinator, generally named as the server, is in charge of aggregating models (or gradients) from local participants and sending back the aggregated values. For example, a typical way to combine local models is the "Model Averaging Algorithm" for such an aggregation step:

$$w^{t+1} = \frac{1}{m}\sum_{i=1}^{m} w_i^t, \tag{1}$$

where w_i^t denotes the weight parameters of the i-th participant on the t-th iteration, and w^t represents the weight parameters of the global model at the t-th iteration that acts as the start point for the next round of local training.

An alternative way is to update local gradients to the server and then perform a global gradient descend step to obtain the model for the next round:

$$\nabla g^t = \frac{1}{m}\sum_{i=1}^{m} \nabla g_i^t, \tag{2}$$

$$w^{t+1} = w^t - \eta \nabla g^t. \tag{3}$$

2.2 Multi-Party Computation

Multi-party computation (MPC) represents a conventional encryption methodology with the goal of creating mechanisms for parties to jointly compute a function while keeping each individual part safe. Generally, the basic properties that an MPC protocol should ensure include:

- *Lossless encryption*: The encryption of local information should not sacrifice the overall correctness.
- *Privacy protection*: No information about the private data held by the parties can be inferred from the messages sent during the execution of the protocol.

Federated learning requires multiple parties to jointly optimize some predefined loss function while expecting local data to be secure. As such, the MPC tends to be a good candidate when local parties are concerned about the safety of data and models. In general, FL algorithms utilize the MPC in the following order:

1) Local party splits its true model and sends the fraction to neighbors.
2) The received fraction is combined with each party's own residual model to create a useless new model.

The rest steps follow the general federated learning scheme by uploading local models to the server and then receiving the combined model for the next round update.

2.3 Concerns of Applying MPC to Federated Learning

The general belief for federated learning is that by sending gradients or models, instead of the data itself, the underlying sensitive data can be utilized only locally and kept private. But yet, recent studies [10], [11], [25] indicate that image recovery from gradients is possible by starting from an arbitrary image-label pair (x', y') and then minimizing the gradient difference. This leads to a backdoor for conventional MPC-based FL research: though the server cannot obtain each local gradient ∇g_i precisely, it can get the true average gradient ∇g^t for each round. Note each local gradient is the average of multiple images (e.g., N images), then the server can consider the combined gradient ∇g^t as the average value of a larger batch (e.g., mN images) and conduct reverse engineering to recover images.

2.4 Augmented MPC with Two-Round Model Decomposition

To eliminate the possibilities of reconstructing the local models and deny access to the true global model for the server, we provide augmentation to the standard MPC so that stronger protections are offered. The method we propose is to further split local shares into a public share for transmission and a secret share to stay locally, so that the server can only receive randomly biased local models. For simplicity, we consider the model averaging algorithm for backbone FL aggregation method, though the following method also applies to the gradient updating method.

2.4.1 First-Round Model Decomposition

Specifically, each local participant (or named as a "node" in graph theory) first decomposes its model into m secret shares:

$$w_i^t = \sum_{j=1}^{m} w_i^{j,t} \quad \text{for} \quad i \in [1, 2, \cdots, m]. \tag{4}$$

Throughout this paper, the subscripts represent the origin of the model, whereas the superscripts denote the destination and round number. For example, the above formula can be read as: model w_i^t of participant i will be decomposed into m pieces, with $w_i^{j,t}$ representing a model piece to be transmitted from participant i to participant j on the t-th round.

The intuition behind such a model decomposition is that rather than having a single model handled by a single participant, it would be generally more secure to decompose the model into multiple pieces and allow multiple parties to manage these pieces. Any single fraction of $w_i^{j,t}$ often does not present any useful information of local knowledge.

Algorithm 1 Augmented MPC with Two-Round Model Decomposition for Secure Federated Learning

```
Local participant $i = 1, 2, \cdots, m$:
1:  Generate public keys $(n_i, e_i)$ and private key $d_i$
2:  for $t = 1, 2, \cdots, T$ do
3:      Learning on the local dataset to obtain local model $w_i^t$
4:      Perform the first-round model decomposition: $w_i^t = \sum_{j=1}^{m} w_i^{j,t}$
5:      Encode seed $c_i^{j,t} = $ RSA.Encode$(\alpha_i^t)$ from node $j$ for $j \in [1, 2, \cdots, m]$
6:      Communicate with neighbours: broadcast $\{w_i^{j,t}, c_i^{j,t}\}$ to node $j$ and receive $\{w_j^{i,t}, c_j^{i,t}\}$ from node $j$ for $j \in [1, 2, \cdots, m]$
7:      Decode seed $\alpha_j^t = $ RSA.Decode$(c_j^{i,t})$ for $j \in [1, 2, \cdots, m]$
8:      Perform the second-round model decomposition: $w_j^{i,t} = \tilde{w}_j^{i,t} \oplus \bar{w}_j^{i,t}$ for $j \in [1, 2, \cdots, m]$ based on $\alpha_j^t$
9:      Aggregate public model: $\tilde{w}_i^t = \sum_{j=1}^{m} \tilde{w}_j^{i,t}$ and private model $\bar{w}_i^t = \sum_{j=1}^{m} \bar{w}_j^{i,t}$
10:     Push public model $\tilde{w}_i^t$ to the server
11:     Receive the global model $\tilde{w}^t$ and reconstruct local model as $w_i^{t+1} = \tilde{w}^t \oplus \bar{w}_i^t$
12: end for
Server:
1:  for $t = 1, \cdots, T$ do
2:      Global model: $\tilde{w}^t = \frac{1}{m} \sum_{i=1}^{m} \tilde{w}_i^t$
3:  end for
```

2.4.2 Second-Round Model Decomposition

After the above decomposition step, each node i proceeds to the local communication step where it transmits its model pieces to neighbouring parties and also receives model pieces $w_j^{i,t}$ from $\forall j \in [1, 2, \cdots, m]$ if the network is fully-connected. These received model pieces will then go through a second-round decomposition:

$$w_j^{i,t} = \tilde{w}_j^{i,t} \oplus \bar{w}_j^{i,t}. \tag{5}$$

Here $\tilde{w}_j^{i,t}$ represents a public model piece to be sent to the server, and $\bar{w}_j^{i,t}$ denotes a randomly bias term as private model piece on local machines. The $\oplus$ decomposition above can be performed in many ways: the local participant can maintain the principal components or simply generates a random local share and then deduct it from the true model. For simplicity, we consider the latter case throughout the paper.

The key step here is to guarantee all neighbours receive the same bias terms so that they can add it back after receiving global model from the server. Namely, the bias terms should be equal for all the local participants:

$$\bar{w}_j^{1,t} = \bar{w}_j^{2,t} = \cdots = \bar{w}_j^{m,t}. \tag{6}$$

A direct approach would be constructing a companion sequence $\bar{w}_j^{i,t}$ while generating the model pieces $w_j^{i,t}$ and deducting it from the latter values. Yet, such an approach forces each participant to transmit two models simultaneously and requires double bandwidth for local communication. Considering the fact that machine learning models, especially deep neural networks, usually consist of a large number of parameters, such an approach leads to a significant amount of extra communication costs.

A potential goal throughout the paper is to augment the original MPC algorithm while minimally triggering extra costs. Therefore, we consider an alternative approach by generating a unique key α_j^t on node j, and then transmit such a key to its neighbours indirectly. With this key, Equation (6) would hold for all participants. In terms of practical implementations, such a key can be a simple "seed" when generating random sequences. For example, in Numpy and PyTorch, the bias term $\bar{w}_j^{i,t}$ can be generated by setting the seed as:

$$\text{numpy.random.seed}(\alpha_j^t) \;\text{ or }\; \text{torch.manual\_seed}(\alpha_j^t).$$

In these cases, the seed α_j^t (or the key) is generally only a *scalar*, thus the extra cost during model transmission can be almost neglected when compared to the size of the original model piece $w_j^{i,t}$. For privacy concerns, this seed will not be transmitted directly but first be encrypted locally based on the RSA algorithm [26], [27]. This would require each node to possess some public keys (n_i, e_i) and private keys d_i, as standard RSA algorithm demands, and each node will encode such its seed based on its neighbours' public keys:

$$c_i^{j,t} = \text{RSA.Encode}(\alpha_i^t) \quad \forall j \in [1, 2, \cdots, m].$$

Focusing on participant i, after it receives the secret shares $w_j^{i,t}$ and encrypted scaler $c_i^{j,t}$ from its neighbour j, it will then perform a RSA decoding step to recover the true seed α_j^t. With this seed, the second-round model decomposition can be performed as:

$$w_j^{i,t} = \tilde{w}_j^{i,t} + \bar{w}_j^{i,t} \;\text{ for }\; j \in [1, 2, \cdots, m], \tag{7}$$

so that the model piece is shattered again into a public model piece $\tilde{w}_j^{i,t}$ and a private model piece $\bar{w}_j^{i,t}$.

Note the above step is not adding the noise $\bar{w}_j^{i,t}$ to $w_j^{i,t}$ as the differential privacy (DP) approach, and the differences lie in two aspects: 1) the bias term $\bar{w}_j^{i,t}$ can be arbitrary large while in DP the noise needs to be controlled within certain levels; 2) the local participant can later fully reconstruct the underlying true global model by adding the private models back while in DP it is generally impossible for any participant to achieve this goal.

The next step would be aggregating the public and private model pieces:

$$\tilde{w}_i^t = \sum_{j=1}^m \tilde{w}_j^{i,t}; \quad \bar{w}_i^t = \sum_{j=1}^m \bar{w}_j^{i,t},$$

and transmitting $\tilde{w}_i^t$ to the server while keeping $\bar{w}_i^t$ on local machines.

2.4.3 Global Aggregation and Local Reconstruction

The server only has access to the public model $\tilde{w}_i^t$ for all nodes i, where it conducts a simple model aggregation step as Eq (1):

$$\tilde{w}^t = \frac{1}{m} \sum_{i=1}^m \tilde{w}_i^t. \tag{8}$$

Since the private models are not shared with the server, such an aggregated step on the server only grants it a randomly biased global model. This is in contrast to the standard MPC, where the global model may resemble the local models in Figure 1. Later analysis in Theorem 3 will prove that it would be almost impossible for the server to obtain an ϵ-close model.

The last step for the augmented MPC algorithm would be adding the private bias back to reconstruct the underlying true global model:

$$w_i^{t+1} = \tilde{w}^t \oplus \bar{w}_i^t \;\text{ for }\; i \in [1, 2, \cdots, m], \tag{9}$$

so that the true underlying aggregated model can be fully reconstructed: $w_i^{t+1} = \frac{1}{m}\sum_{j=1}^m w_j^t$ for all $i \in [1, \cdots, m]$. Note the above $\oplus$ should be consistent with the operation in Eq (5): for example, if the local participants use the simple arithmetic plus in Eq (5), the above equation would use a simple sum here.

Algorithm 1 summarizes the main steps of our proposed augmented MPC algorithm.

2.5 Summary

Finally, it is necessary to conclude the above algorithm by examining our gains and payments. By augmenting the classical MPC algorithm, we obtain a more secure updating scheme for federated learning that denies the server any access to useful local knowledge. Moreover, this allows us to fully eliminate the local information leakage while being easier to nominate an aggregation center in practice. What the algorithm pays for our gain in providing stronger model protections is only some extra scalar transmission among participants and one additional model decomposition step, which tends to be minimal. Table 1 provides a complete comparison between our proposed method and other encryption methods.

2.6 Extension for Handling Other Graphs

In the above scenario, we consider the situation of a fully-connected graph (or complete graph in graph theory [28]) so that all participants have direct access to the rest nodes. In case the participants do not form a fully connected graph, some participants may not be able to communicate with some nodes directly. Moreover, denote the collections of i-th node's direct neighbors as $N(i)$ (the node itself is considered to be a neighbor), then $N(i)$ can even be dynamic for different round t if participants join and leave during the federated learning. In case not all participants are connected to each other, a few modifications need to be made in order to ensure $w_i^{t+1} = \frac{1}{m}\sum_{j=1}^m w_j^t$.

First of all, model w_i^t needs be decomposed into $\#N(i)$ parts instead of m parts. The communication among participants with direct connections is performed as Algorithm 1, but for those participants not directly connected to node i, the public key (or the seed) in the RSA algorithm needs to be encrypted and transmitted to the server first. The i-th node then receives and decodes such a message with its private key, following by generating a bias term $\bar{w}_j^{i,t}$ and subtracting this term from its own model:

$$w_i^{i,t} = \tilde{w}_i^{i,t} + \left(\bar{w}_i^{i,t} + \sum_{\substack{j=1 \\ j \notin N(i)}}^m \bar{w}_j^{i,t} \right). \tag{10}$$

Note in the above equation, we require the node to add the private parts of non-direct neighbour $j \notin N(i)$ to its own counterpart $\bar{w}_i^{i,t}$.

So the overall bias term (private model) on node i will be

$$\bar{w}_i^t = \sum_{\substack{j=1; j \neq i; \\ j \in N(i)}}^m \bar{w}_j^{i,t} + \bar{w}_i^{i,t} + \sum_{\substack{j=1 \\ j \notin N(i)}}^m \bar{w}_j^{i,t} = \sum_{j=1}^m \bar{w}_j^{i,t}. \tag{11}$$

The rest steps are similar to Algorithm 1.

Algorithm	hide LM for participants	hide LM for server	hide GM for server	fully recover GM for participants
FedLearn	✓	✗	✗	✓
FedLearn with MPC	✓	✓	✗	✓
FedLearn with DP	✓	✓	✓	✗
FedLearn with AMPC	✓	✓	✓	✓

TABLE 1: Properties of various encryption methods. LM refers to the local model and GM refers to the true global model.

3 Theoretical Analysis

We propose an augmented MPC mechanism in which two model decomposition steps are performed locally. It is expected to inherit the merits of standard MPC with minimal extra costs while providing additional model protections simultaneously. These properties shall be validated through theoretical analysis in the following part.

3.1 Lossless Encryption and Decryption

One of the key properties of standard MPC is that model encryption and decryption is performed in a lossless way so that participants can fully recover the true global model w^t. Therefore, before moving towards any extra merits of the augmented MPC algorithm, let us first ensure our gains do not offset this lossless property of MPC.

Theorem 1. *By performing two-round model decomposition and adding bias terms locally, each local participant can fully reconstruct the ground-truth in Algorithm 1, namely $w_i^{t+1} = w^t \quad for \quad i \in [1, 2, \cdots, m]$.*

Proof: We compute the aggregated global model on the server as:

$$\begin{aligned}\tilde{w}^t &= \frac{1}{m}\sum_{i=1}^{m}\tilde{w}_i^t \\ &= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{m}\tilde{w}_j^{i,t} \\ &\overset{(9)}{=} \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{m}\left(w_j^{i,t} - \bar{w}_j^{i,t}\right) \\ &= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{m} w_j^{i,t} - \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{m}\bar{w}_j^{i,t} \\ &\overset{(4)(6)}{=} \frac{1}{m}\sum_{j=1}^{m} w_j^t - \sum_{j=1}^{m}\bar{w}_j^{i,t} \\ &\overset{(1)}{=} w^t - \sum_{j=1}^{m}\bar{w}_j^{i,t}.\end{aligned}$$

In the above proof, we use the fact that the biased terms $\bar{w}_j^{i,t}$ are equal for all $j \in [1, \cdots, m]$ since they are generated with the same seed α_i^t.

Then the reconstructed model on participant i would be $w_i^{t+1} = \tilde{w}^t + \bar{w}_i^t = w^t - \sum_{j=1}^m \bar{w}_j^{i,t} + \sum_{j=1}^m \bar{w}_j^{i,t} = w^t$. This concludes the proof. □

Remark: The server in Algorithm 1 can now have access to the biased global model $\tilde{w}^t$, while the local client is able to obtain the true global model w^t by adding local residuals back. As such, the local model $\{w_i^{t+1}\}$ is now unbiased.

3.2 Privacy-Preserving Learning

The above theorem says our augmented MPC inherits the merits of standard MPC by encrypting and decrypting models in a lossless way, and now we can proceed to the privacy-preserving part. Our considerations include both protections over local models among neighbours and the server, as well as avoiding the true global model w^t to be revealed by the server. Proof for the former claim will be trivial since local models are split into multiple pieces in the first model decomposition step (as standard MPC), and therefore we shall focus on justifying the latter claim.

Lemma 2. *Suppose elements of the bias term $\bar{w}_j^{i,t}$ are generated based on a distribution $\mathcal{F}_j^t$ with the mean μ_j^t and the variance $(\sigma_j^t)^2$, then components of the overall biased term $w^t - \tilde{w}^t$ would obey a distribution with mean and variance as*

$$\bar{\mu}^t = \sum_{j=1}^{m}\mu_j^t, \quad (\bar{\sigma}^t)^2 = \sum_{j=1}^{m}(\sigma_j^t)^2. \tag{12}$$

Proof: The proof is straightforward by considering $w^t - \tilde{w}^t = \sum_{j=1}^m \bar{w}_j^{i,t}$. Note that the components $\bar{w}_j^{i,t}$ in the sum value is generated with its own seed α_j^t so that they are independent across different iterations. The mean and variance of their sum can therefore be easily proved. □

The above lemma paves a road to prove the aggregated public model $\tilde{w}^t$ is sufficiently far from the underlying ground-truth w^t on each round. For simplicity of notations, let us assume $\bar{w}_j^{i,t}$ is non-negative and models $\tilde{w}^t$ and w^t are reshaped into a long vector with N parameters, then the model difference can be computed as follows.

Theorem 3. *The possibility of obtaining an ϵ close model $\tilde{w}^t$ on the server side is*

$$\mathbb{P}\left(\frac{\|\tilde{w}^t - w^t\|_1}{N} < \epsilon\right) < \frac{(\bar{\sigma}^t)^2}{N(\bar{\mu}^t - \epsilon)},$$

for any $\epsilon < \bar{\mu}^t$.

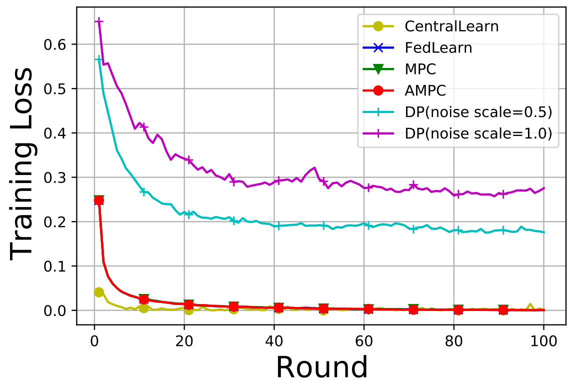
Remark: For deep neural networks, the parameter number N of is generally sufficiently large, so that the above theorem says it would almost be impossible to obtain an ϵ-close model on the server side.

Proof: For the k-th component of $w^t - \tilde{w}^t$, we denote its absolute value as

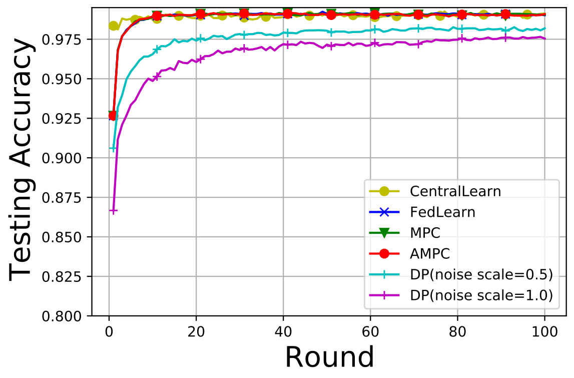
$$\beta_k = |[w^t]_k - [\tilde{w}^t]_k| = \sum_{i=1}^{m}[\bar{w}_i^{j,t}]_k.$$

Lemma 2 ensures each component β_k is generated independently with mean $\mathrm{E}[\beta_k] = \bar{\mu}^t$ and variance $\mathrm{Var}[\beta_k] = (\bar{\sigma}^t)^2$. As consequences, for any $\epsilon < \bar{\mu}^t$, we have

$$\begin{aligned}&\mathbb{P}\left(\frac{\sum_{k=1}^N \beta_k}{N} < \epsilon\right) \\ &< \mathbb{P}\left(\left|\frac{\sum_{k=1}^N \beta_k}{N} - \bar{\mu}^t\right| > \bar{\mu}^t - \epsilon\right) \\ &\le \frac{(\bar{\sigma}^t)^2}{N(\bar{\mu}^t - \epsilon)},\end{aligned}$$

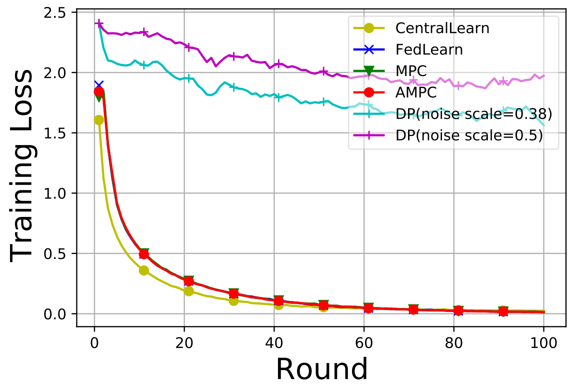


(a) Training loss on the MNIST dataset.

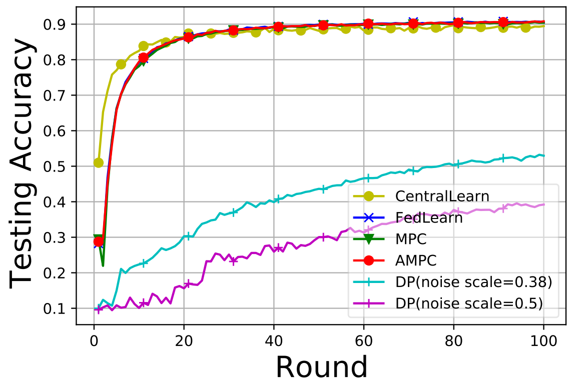


(b) Testing accuracy on the MNIST dataset.

Fig. 2: Performance of LeNet-5 on the MNIST dataset. Samples are randomly distributed to 10 participants.



(a) Training loss on the CIFAR-10 dataset.



(b) Testing accuracy on the CIFAR-10 dataset.

Fig. 3: Performance of VGG-16 on the CIFAR-10 dataset. Samples are randomly distributed to 10 participants.

where in the last step we use Chebyshev inequality. By the definition of ℓ_1-norm on vectors, we have $\|\tilde{w}^t - w^t\|_1 = \sum_{k=1}^N \beta_k$ and conclude the proof. $\square$

In Lemma 2 and Theorem 3, we consider the general case and do not specify how the bias terms $\bar{w}_j^{i,t}$ are generated. To give a simple example, we consider the case where the bias terms $\bar{w}_j^{i,t}$ are generated based on a uniform distribution in the following corollary.

Corollary 4. *Suppose the biased terms are generated based on a uniform distribution, namely $\bar{w}_j^{i,t} \sim \mathbb{U}[0,1]$, then we have*

$$\mathbb{P}\left(\frac{\|\tilde{w}^t - w^t\|_1}{N} < \epsilon\right) < \frac{m}{6N(m-2\epsilon)} \approx \frac{1}{6N},$$

for $\epsilon \ll m$.

Remark: N refers to the parameter number of the backbone models, for instance millions in deep neural network. Hence the possibility to recover the true model is almost 0.
Proof: Proof is straightforward by plugging the mean and variance of uniform distribution, hence omitted. $\square$

4 Experimental Study

Having established theoretical guarantees, our next step would be testing its actual performance on real-world datasets. Our goal in this part is still two-fold: showing AMPC algorithm obtains lossless encryption and decryption by examining the training curves; showing it provides stronger security protections by measuring the difference between local models and the (biased) global model.

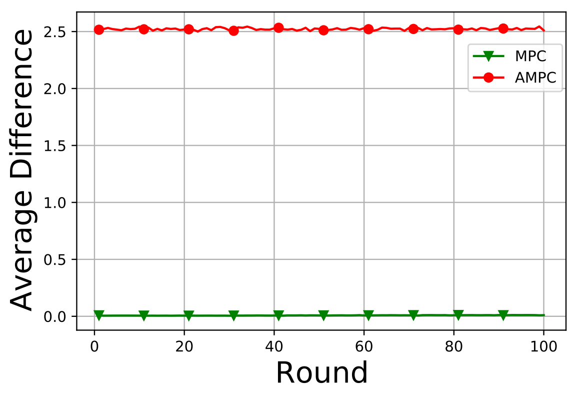
4.1 Experimental settings

Performance of the algorithms are reported on two public datasets MNIST [29] and CIFAR-10 [30], together with two neural architectures LeNet-5 and VGG-16 [31]. Specially, the following algorithms are considered:

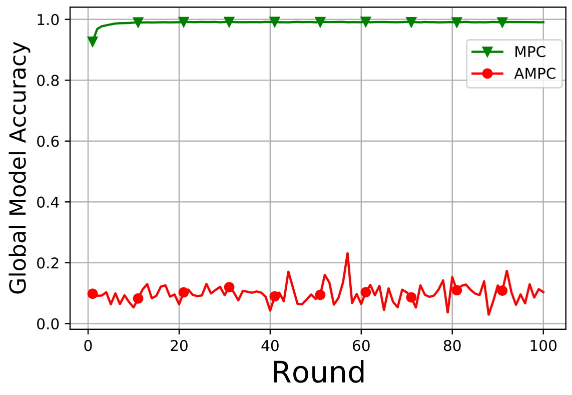
1) **CentralLearn**: centralized learning by sending all data to the server and updating only the global model.
2) **FedLearn**: federated learning without performing MPC or DP.
3) **MPC**: standard MPC algorithm from PySyft.
4) **DP**: Renyi differential privacy algorithm [32] [1] by performing RDP on local models. Noise levels and delta (δ) are set as suggested values and the clipping threshold of the gradient norm is set as 1.
5) **AMPC**: the proposed algorithm in this paper.

We use Adam [33] as the network optimizer, with its batch size set as 128 and learning rate fixed at 10^{-3}. A federated learning

1. Source code from Opacus:https://github.com/pytorch/opacus.

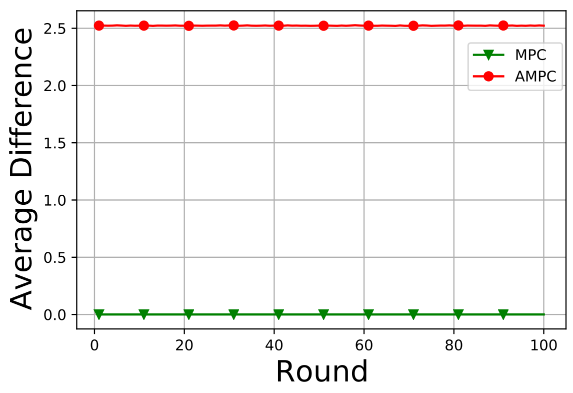


(a) Global model difference.

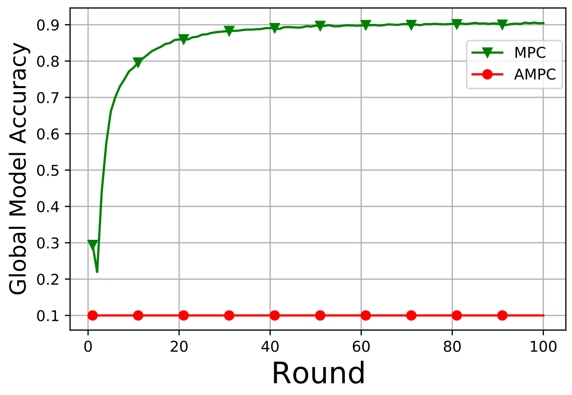


(b) Testing accuracy with global model on server.

Fig. 4: The left figure represents the model difference $\frac{\|\tilde{w}^T - w^T\|_1}{N}$ between the aggregated model on the server and true global model. The right figure shows the testing accuracy by utilizing the model on the server. Experiments are reported on the MNIST dataset.



(a) Global model difference of VGG-16



(b) Testing accuracy with global model on server.

Fig. 5: The left figure represents the model difference between the aggregated model on the server and true global model. The right figure shows the testing accuracy by utilizing the model on the server. Experiments are reported on the CIFAR-10 dataset.

system is generated with 10 local participants and 1 server, with each local participant receiving random data from the dataset and performing 5 inner rounds of local updates.

4.2 Convergence and Prediction Accuracy

For privacy concerns, local models of federated learning will generally go through certain encryption steps and then decrypted after model aggregation. In the meanwhile, we expect these steps will have minimal effects on the convergence rate during training, and also the final prediction accuracy.

Figure 2 and 3 report the training and testing performance of various algorithms on these two datasets. As shown in the figure, the curves of the MPC-based algorithms are most identical to the non-encrypted federated learning (FedLearn), representing the fact that models are fully reconstructed so that these additional MPC encryption and decryption steps do not affect the final performance. Note such a conclusion is true for both MPC and AMPC, since the latter is designed to inherit the lossless property of standard MPC as shown in Theorem 1.

In contrast to lossless property of AMPC is the accuracy drops of DP algorithm. By adding noise locally, the central server can only obtain some perturbed local models for model aggregation. In the Opacus framework, such a perturbation leads to 1.1% and 2.0% accuracy drops with noise scale set as 0.5 and 1 on the MNIST dataset, and almost 30% accuracy drops on the CIFAR-10 dataset. As references for results on the CIFAR-10 dataset, readers can refer to similar reported in Opacus[2] where the accuracy of centralized learning with DP-based ResNet-18 drops to 56%, also see other centralized results in [34], [35]. Note here we do not perform an exhaustive search for hyper-parameters of DP, either using default or previously published values.

4.3 Model Difference and Privacy Preserving

The MPC-based algorithms can prevent the model from suffering from accuracy drops, but as shown in Figure 1, possibilities still exist as the server or third-parties can obtain a close model to local counterparts. We enhance the security protection by augmenting the MPC algorithm so that only local participants can obtain the true underlying global model in Theorem 3. In this part, we shall validate such a conclusion with practical experiments.

Figure 4(a) illustrates the model difference $\frac{\|\tilde{w}^t - w^t\|_1}{N}$ on the MNIST dataset. For the MPC algorithm, the difference is exactly zero since no biased term is introduced. In practice, this leads

2. https://opacus.ai/tutorials/building_image_classifier

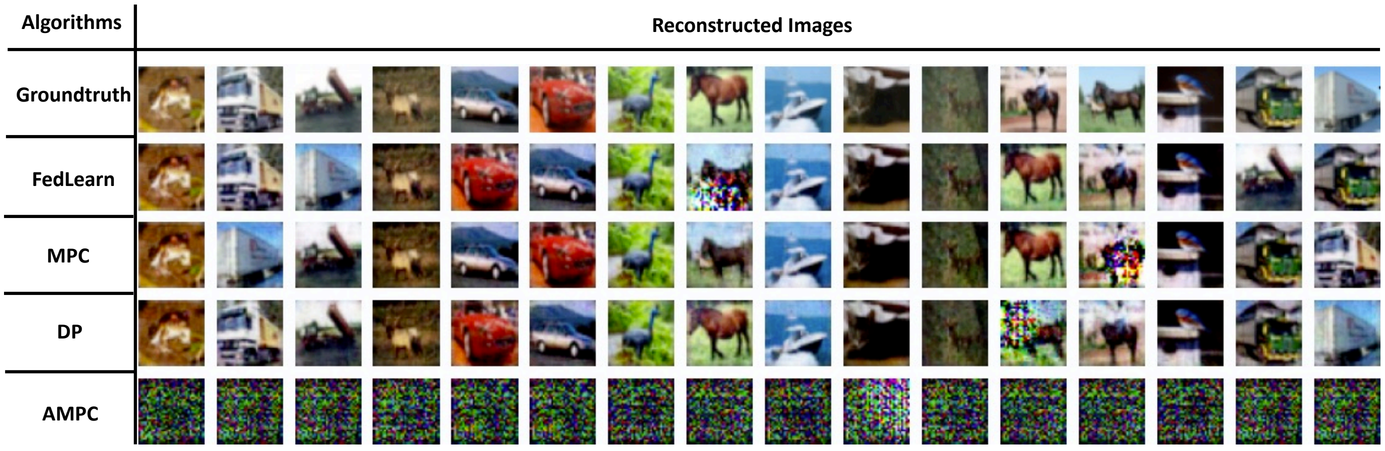


Fig. 6: Defence against gradient leakage attack for various algorithms. The first row represents the ground-truth images, and the rest lines show the obtained images by gradient inverse method for 2000 iterations [36].

to a situation where we are forced to select a trusted third-party for all local participants and exposed to possibilities of leaking such an aggregated model during transmission. Such a situation can be avoided for AMPC algorithm, as the aggregated model is sufficiently far from the underlying ground-truth on all rounds (Figure 4(a)). Moreover, if the server tries to utilize the aggregated model or the model is leaked to any third-party, the final performance on the MNIST dataset is only around 10% accuracy, as shown in Figure 4(b). Note the dataset contains 10 classes, so it represents a fact that prediction with public model alone only provides random guess. As comparisons, the averaged model obtains good performance for the MPC algorithm. Similar results are reported in Figure 5 on the CIFAR-10 dataset.

To summarize, the proposed MPC mechanism can protect local participants' privacy by only transmitting biased public models to the server for aggregation, and then fully reconstructing the underlying true global model. In practice, this represents an ideal situation where we are able to protect model privacy without sacrificing the overall learning accuracy.

4.4 Defence Against Gradient Leakage

As alluded to earlier, the gradient inverse method [10] allows the malicious attacker to recover the underlying images by performing gradient inverse engineering. While their experiments are conducted on the FedAvg algorithm, we notice conventional MPC method could also face this potential issue since the server can obtain an unbiased gradient/model after aggregation.

To validate, we follow the SOTA deep leakage method [36] to utilize the generative model for gradient inverse engineering. Specifically, we use 4 local clients to build a whole batch size of 16 images on the CIFAR-10 dataset. Results (Figure 7) indicate that the server (or any malicious attacker) could start from some random noise and then gradually match the obtained gradients to the truth gradients to recover the underlying true pictures. For all algorithms except for AMPC, the structural similarities (SSIM) between the inverse images and the ground-truth continue to increase, whereas the SSIM keeps low for the proposed AMPC algorithm.

Figure 6 also shows the final obtained images after 2000 rounds, as well as the ground-truth for comparison. For the unprotected FedAvg and the conventional MPC algorithm, the server could almost identically obtain all the images when the batch size

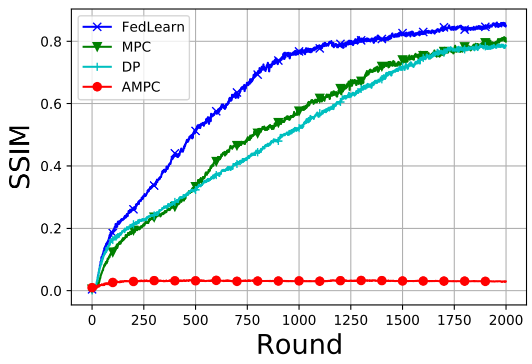


Fig. 7: Image quality assessment for various algorithms during the gradient inverse. The vertical axis SSIM represents the structural similarity between the inverse images and the ground-truth.

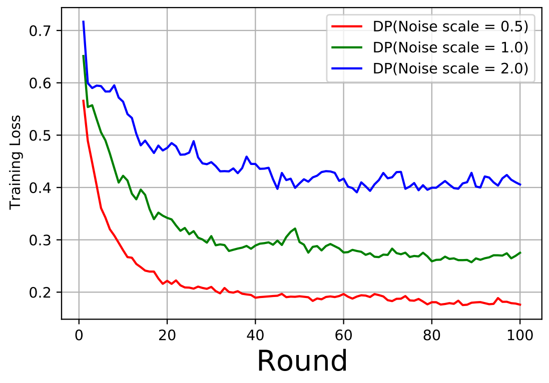
equals 16. The DP algorithm provides different levels of noise-adding before sending the model to the server, hence the obtained image quality by gradient inverse tends out to be lower than the previous two algorithms. But we also show that under minor noise level ($N(0, 10^{-4})$), obtaining high-quality images is still possible. In contrast, by providing a two-round MPC, the server in AMPC algorithm can only receive a biased model/gradient, hence the gradient matching is no longer possible. After 2000 local iterations, the server can only obtain some random noise in our experiment, hence providing more secure defences against the gradient inverse methodology.

4.5 Additional Cost

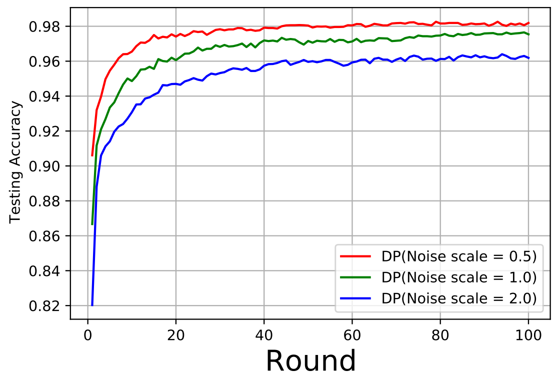
Finally, it is worth mentioning the additional time and communication costs led by AMPC: 1) the coding, decoding and model decomposition steps can be processed within seconds on the tested GPU (2080Ti), hence the extra time cost tends to be minor when compared to other algorithms; 2) the AMPC algorithm triggers additional peer-to-peer communication costs when compared to FedAvg and DP algorithm, but its communication load is generally on the same level as the conventional MPC algorithm.

5 Conclusion

The goal of this paper is to propose an effective method (AMPC) for secure federated learning systems. By observing the difference



(a) Training Loss for DP



(b) Testing accuracy for DP.

Fig. 8: Performance of DP algorithm with various noise levels on the MNIST dataset.

of the global model and local models, we reveal the fact that standard MPC could fail in certain scenarios. We therefore introduce a two-step decomposition for the secret shares, and render the true global model invisible to the central server, which also leads to the simplicity of selecting a "server" in practice. A series of theoretic analysis are provided to illustrate the effectiveness of the proposed method, along with practical experimental results on the MNIST and CIFAR-10 datasets.

6 Additional Proof and Experiment

6.1 Proof for Other Graphs

Similar to Theorem 1, the MPC encryption and decryption for general graph is also performed in a lossless way.

Theorem 5. *Suppose all the participants form a graph $\mathcal{G}$ and the bias terms are generated according to Equation (10), then the local participant can reconstruct the true global model:*

$$w_i^{t+1} = w^t \ for\ i \in [1, 2, \cdots, m].$$

6.1.0.1 Proof: Proof will be similar by considering the averaged global model first:

$$\begin{aligned}
\tilde{w}^t &= \frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1;\\ j\in N(i)}}^{m} \tilde{w}_j^{i,t} \\
&= \frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1;j\neq i;\\ j\in N(i)}}^{m} \left(\tilde{w}_j^{i,t}\right) + \frac{1}{m}\sum_{i=1}^{m}\tilde{w}_i^{i,t} \\
&\overset{(10)}{=} \frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1;j\neq i;\\ j\in N(i)}}^{m} \left(w_j^{i,t} - \bar{w}_j^{i,t}\right) \\
&\quad + \frac{1}{m}\sum_{i=1}^{m}\left(w_i^{i,t} - \bar{w}_i^{i,t} - \sum_{\substack{j=1\\ j\notin N(i)}}^{m} \bar{w}_j^{i,t}\right) \\
&= \frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1;\\ j\in N(i)}}^{m} w_j^{i,t} - \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{m}\bar{w}_j^{i,t}
\end{aligned}$$

$$= w^t - \sum_{j=1}^{m}\bar{w}_j^{i,t}.$$

Then the local model on node i would be

$$w_i^{t+1} = \tilde{w}^t + \bar{w}_i^t \overset{(11)}{=} w^t - \sum_{j=1}^{m}\bar{w}_j^{i,t} + \sum_{j=1}^{m}\bar{w}_j^{i,t} = w^t.$$

This concludes the proof. □

The proof for stronger privacy-preserving is exactly the same as Theorem 3 and hence omitted.

6.2 DP Experiments on MNIST

We have provided results of the DP algorithm on MNIST dataset in Figure 2(a) and 2(b), with noise level set as 0.5 and 1.0. But DP algorithm is very sensitive to the noise added, and user often has to face trade-off between privacy level (hence noise level) and the final performance. We provide an additional experiment by providing a larger noise level 2.0 in the following figure. Results are consistent with our previous observations that the noise level incurs additional performance loss in DP algorithm.

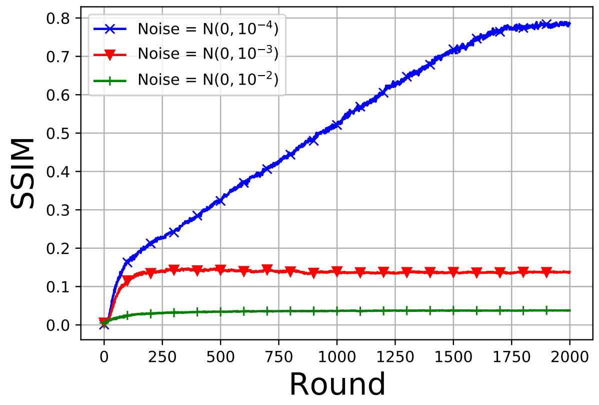


Fig. 9: The obtained images from gradient inverse with different DP noise levels. Here N represents the normal distribution.

6.3 Gradient Inverse with Various DP Noise

DP algorithm allows the local client to add different levels of noise to the model before sending it to the server. In the previous section, we consider a minor noise case, while detailed experimental results are presented in this subsection. Figure 9 indicates the increasing noise level could reduce the obtained image quality from gradient inverse algorithm. Note in general, the DP algorithm does not utilize a fixed noise level but gradually increases the noise from low to high during the training process. Higher noise level can provide better model protection, while in the same time lead to inaccurate model aggregation, in contrast to the lossless properties of MPC and AMPC.

REFERENCES

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

[2] 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.

[3] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for iot devices," *IEEE Internet of Things Journal*, 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2020.3017377

[4] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *CoRR*, 2015. [Online]. Available: arXiv:1511.03575

[5] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, 2016. [Online]. Available: arXiv:1610.05492

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[8] C. Zhang, Q. Li, and P. Zhao, "Decentralized optimization with edge sampling." in *IJCAI*, 2019, pp. 658–664.

[9] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," *CoRR*, vol. abs/1802.08232, 2018. [Online]. Available: https://arxiv.org/abs/1802.08232

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

[11] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," *CoRR*, 2020. [Online]. Available: arXiv:2001.02610

[12] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation*. Springer, 2008, pp. 1–19.

[13] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[14] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "Ldp-fed: Federated learning with local differential privacy," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, 2020, pp. 61–66.

[15] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 1982, pp. 160–164.

[16] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.

[17] 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.

[18] 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 *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE, 2020, pp. 410–419.

[19] J. Wang, D. He, A. Castiglione, B. B. Gupta, M. Karuppiah, and L. Wu, "Pcnncec: Efficient and privacy-preserving convolutional neural network inference based on cloud-edge-client collaboration," *IEEE Transactions on Network Science and Engineering*, 2022.

[20] E. Sotthiwat, L. Zhen, Z. Li, and C. Zhang, "Partially encrypted multi-party computation for federated learning," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2021, pp. 828–835.

[21] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[22] C. Zhang and Q. Li, "Distributed optimization for over-parameterized learning," *arXiv:1906.06205*, 2019.

[23] C. Zhang and L. Qianxiao, "Distributed optimization for degenerate loss functions arising from over-parameterization," *Artificial Intelligence*, vol. 301, p. 103575, 2021.

[24] L. Lyu, X. Xu, Q. Wang, and H. Yu, "Collaborative fairness in federated learning," in *Federated Learning*. Springer, 2020, pp. 189–204.

[25] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 937–16 947, 2020.

[26] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[27] C. C. Cocks, "A note on non-secret encryption," *CESG Memo*, 1973.

[28] J. A. Bondy, U. S. R. Murty *et al.*, *Graph theory with applications*. Macmillan London, 1976, vol. 290.

[29] 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.

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

[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] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 263–275.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, 2014. [Online]. Available: arXiv:1412.6980

[34] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[35] A. Triastcyn and B. Faltings, "Bayesian differential privacy for machine learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9583–9592.

[36] J. Jeon, K. Lee, S. Oh, J. Ok *et al.*, "Gradient inversion with generative image prior," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 898–29 908, 2021.