SentinelFL: Noise-Driven Global Model Protection Against Model Theft in Federated Learning

Daeho Kwon
Department of Mathematics
Soongsil University
Seoul, Republic of Korea
kwndh01@soongsil.ac.kr

Bong Jun Choi
School of Computer Science and Engineering
Soongsil University
Seoul, Republic of Korea
davidchoi@soongsil.ac.kr

***Abstract*—In federated learning, model theft by malicious clients poses a significant risk to the intellectual property of the model owner. However, the central server must share the model parameters with clients for training, making it vulnerable to such attacks. This study proposes a mechanism to mitigate the risk of model theft by adding calculated noise to the model parameters, ensuring that the shared model retains low accuracy to prevent theft while preserving high accuracy for the global model created from aggregated local models. The proposed method introduces a Noise Control Unit that minimizes the loss of accuracy for the global model while significantly reducing the chances of model theft. Experimental results show that the proposed approach successfully maintains high global model accuracy and protects model privacy. This work contributes to enhancing privacy in federated learning frameworks, offering a viable solution to balance model privacy and performance.**

***Index Terms*—federated learning, model theft attack, differential privacy, crowdsourcing**

I. Introduction

Since the emergence of Federated Learning (FL) [1], its adoption has grown across various training scenarios. Recently, in particular, FL-based crowdsourcing platform [2], [3] encourages client participation, providing a means for organizations that cannot independently build models to commission specialized model training companies. However, it raises intellectual property concerns, as the model requester and the platform are separate entities.

While the crowdsourcing platform must share the model parameters with clients in each round of training under the traditional FL scenarios, this creates a potential risk of model theft by malicious clients. To mitigate this issue, existing methods such as homomorphic encryption [4], [5] have been proposed. However, these methods are often related to excessive time complexity and high computational cost on the client's local device.

In response, this study proposes a novel mechanism that intentionally adds noise to the model parameters, thus degrading their utility for malicious clients while ensuring that the model provided to the model requester retains its performance. The goal of this research is to strike a balance between maintaining model privacy in FL environments and preserving the performance of the model, addressing the challenges of both privacy protection and model accuracy.

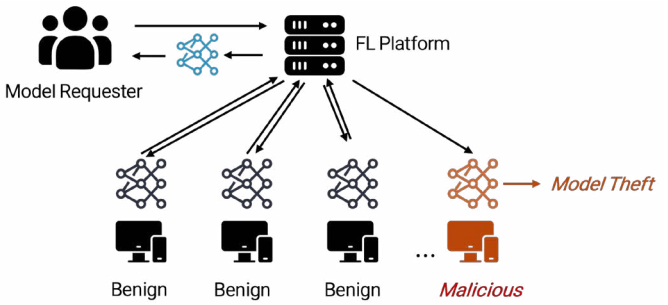


Fig. 1: Model theft in FL-based crowdsourcing platform; the global model shared by the FL platform can be easily copied or reconstructed by the clients

TABLE I: Comparison of centralized learning, federated learning, and SentinelFL

	Centralized	**Federated**	**SentinelFL (Proposed)**
Data privacy	**No**; data collected at server	**Yes**; data kept locally	**Yes**; data kept locally
Model privacy	**Yes**; model not shared	**No**; parameters shared	**Yes**; noisy parameters shared
Attack mechanism	Model extraction	(1) Model theft (Insider), (2) Extraction (Outsider)	(1) Model theft (Insider), (2) Extraction (Outsider)

II. Related Works

A. Model Theft Attack

Model Theft [6] refers to an attack mechanism where the model is copied or replicated without authorization. The strategies employed to mitigate model theft in FL differ significantly from those used in traditional (centralized) learning frameworks. In centralized learning, the model architecture is typically concealed from the clients, with clients only transmitting their local data to a central server. As a result, defensive strategies in centralized environments place less emphasis on protecting against the direct theft of model parameters than on other threats (e.g., Model Extraction Attack [7], [8]).

In contrast, FL operates in a more transparent environment, where model parameters are exposed to clients during the training process. This transparency provides clients with greater visibility into the model's internal structure but also

increases the risk of exploitation by malicious clients. Given that clients can interact with and observe the model, there is a heightened potential for unauthorized replication or theft. This distinction emphasizes the need for robust defensive mechanisms tailored to mitigate the risks of model theft in FL. To address these concerns, it is essential to enhance technical safeguards that effectively counter malicious actions while ensuring the privacy preservation of the model while maintaining its performance.

B. Differential Privacy

Differential Privacy (DP) [9], [10] is a robust data analysis technique that provides a quantifiable privacy guarantee for individual data within a dataset. This guarantee is formalized through a mathematical condition known as ε-Differential Privacy (ε-DP). An algorithm M is said to satisfy ε-DP if, for any two neighboring datasets D_1 and D_2 that differ by at most **one individual's data**, the probability of any particular outcome occurring remains nearly identical, with the similarity bound controlled by the parameter ε.

Definition 1. (ε-Differential Privacy)
Let $\varepsilon > 0$. A randomized function M is said to be ε-differentially private if, for all neighboring datasets D_1 and D_2 that differ by at most one element, and for all subsets $S \subseteq \text{Range}(M)$, the following condition holds:

$$\frac{\Pr[M(D_1) \in S]}{\Pr[M(D_2) \in S]} \leq e^{\varepsilon}.$$

A smaller ε provides stronger privacy, as the algorithm's output changes minimally when any single individual's data is added, making it difficult to infer their participation while preserving overall dataset insights.

The reason for applying DP to model training is to ensure that the dataset is not exposed during the model training process. As outlined in **Definition 1**, the goal is to protect the data so that specific individuals cannot be identified.

In contrast, in our scenario, the primary objective is to protect the privacy of the global model itself. Unlike traditional scenarios, we should add noise to the global model's parameters. This approach obscures the structure of the global model and diminishes its utility. Additionally, since the server has complete control over how the noise is added to the distributed model and interpreted after collecting the updated model parameters from the client, we can achieve robust privacy control. Therefore, we can devise a **Noise Control Unit** that can control how noise affects the model performance and privacy.

III. Global Model Protection Considerations

To achieve the goal of protecting the global model, we define three conditions that must be satisfied:

- **First**, the client-distributed model should not guarantee *utility*.
- **Second**, the global model should guarantee *utility*.
- **Third**, the computational cost on local devices should remain low, ensuring that the protection mechanisms are economically feasible without excessive resources or time.

At this point, it is necessary to define what we mean by *utility*. The utility of a model can be considered guaranteed if one of the following circumstances holds:

- **First**, the model must be accessible for client use.
- **Second**, if shared or stolen without authorization, the model should not be traceable.
- **Third**, the model should support self-replication, allowing clients to duplicate or modify it without permission.
- **Finally**, the model's practicality (e.g., accuracy, F1 score) must be ensured, with the ability to maintain and improve performance over time.

The goal of this study is to reduce the practicality of the client-distributed model, thereby breaking the final utility condition. While the global model is generally expected to outperform client-distributed models, the key objective here is to degrade the performance of client-distributed models to the extent that they become impractical.

IV. Noise-Driven Global Model Protection

We propose a noise-driven federated averaging method to protect the global model. This approach aims to degrade the accuracy of client-distributed models by introducing noise into the parameters of deep learning models, while simultaneously enhancing the accuracy of the global model. In this study, we use three different global models, unlike conventional Federated Averaging (FedAvg) [1]. These include client-distributed models, CM_1, and CM_2, which function like client-side global models, and global model GM for the model requester.

The basic concept is that the client-distributed models will experience accuracy degradation due to added noise during each round. The noise follows a normal distribution and is applied to the parameters as follows:

$$w_{1,t} \leftarrow w_{1,t} + \mathcal{N}(0, \sigma^2), w_{2,t} \leftarrow w_{2,t} - \mathcal{N}(0, \sigma^2),$$

where $w_{1,t}$ and $w_{2,t}$ are the parameters for CM_1 and CM_2 at round t, respectively. Here, we define σ as the noise intensity value, which is dynamically adjusted in each round to ensure that the accuracy of both CM_1 and CM_2 does not exceed a certain threshold, as discussed later.

The modified parameters are then shared with clients, where $S_{1,t}$ and $S_{2,t}$ denoted the groups receiving CM_1 and CM_2 parameters, respectively.

The parameters from all models in the client groups $S_{1,t}$ and $S_{2,t}$ are aggregated through FedAvg, and the resulting parameters update CM_1 and CM_2. Therefore, at the end of each round, the parameters of CM_1 and CM_2 remain identical. The aggregation method for GM, however, is slightly different. Here, we introduce the concept of a **server-side client**. A server-side client is a client that uses the previous round's GM as its local model, and we define these clients as the server-side client group S_t. Although S_t consists of multiple clients, they essentially represent one server-side client in the form of dummy clients. This server-side client group is added

Algorithm 1: `Noise-Driven Federated Averaging`

Description:
K: total number of clients, indexed by k.
B: local mini-batch size.
E: number of local training epochs.
η: learning rate.
C: fraction of clients participating in each round.
n_k: number of data samples on client k.
P_k: index set of data points on client k.
l: number of server-side clients.
τ: utility guarantee threshold
C_1: number of input channels in CONV1.
H_1, W_1: height and width of the input to CONV1.
K_1: number of filters in CONV1.

Server Execution:
Initialize: w_0
for *each round* $t = 1, 2, 3, \ldots$ **do**
  $m \leftarrow \max(\lfloor C \cdot K \rfloor, 1)$
  $S_{i,t} \leftarrow$ random subset of $\frac{m}{2}$ clients, $i \in \{1, 2\}$
  $S_t \leftarrow$ randomly subset of l clients
  for *each client* $k \in S_t$ ***and*** $t > 1$ **do**
    $w_t^k \leftarrow w_{t-1}$
  $\sigma \leftarrow$ **NoiseControlUnit**$(\delta, \varepsilon, n, \gamma)$
  while $CM_{1,test} \geq \tau$ ***or*** $CM_{2,test} \geq \tau$ ***and*** $t > 3$ **do**
    $u \leftarrow 0$
    if $u = 1$ **then**
      $w_{1,t}^{\text{CONV1}} \leftarrow w_{1,t}^{\text{CONV1}} - \xi$
      $w_{2,t}^{\text{CONV1}} \leftarrow w_{2,t}^{\text{CONV1}} + \xi$
    $\xi \sim \mathcal{N}(0, \sigma^2), \quad \xi \in \mathbb{R}^{C_1 \times H_1 \times W_1 \times K_1}$
    $w_{1,t}^{\text{CONV1}} \leftarrow w_{1,t}^{\text{CONV1}} + \xi$
    $w_{2,t}^{\text{CONV1}} \leftarrow w_{2,t}^{\text{CONV1}} - \xi$
    $CM_{1,\text{test}} \leftarrow \text{Acc}(CM_{1,t})$
    $CM_{2,\text{test}} \leftarrow \text{Acc}(CM_{2,t})$
    $u \leftarrow 1$
  for *each client* $k \in S_{1,t}$ ***in parallel*** **do**
    $w_t^k \leftarrow$ **ClientUpdate**$(k, w_{1,t})$
  for *each client* $k \in S_{2,t}$ ***in parallel*** **do**
    $w_t^k \leftarrow$ **ClientUpdate**$(k, w_{2,t})$
  if $t > 3$ **then**
    $m_t \leftarrow \sum_{k \in S_{1,t} \cup S_{2,t} \cup S_t} n_k$
    $w_{t+1} \leftarrow \sum_{k \in S_{1,t} \cup S_{2,t} \cup S_t} \frac{n_k}{m_t} w_t^k$
  $m_{1,t}, m_{2,t} \leftarrow \sum_{k \in S_{1,t} \cup S_{2,t}} n_k$
  $w_{1,t+1}, w_{2,t+1} \leftarrow \sum_{k \in S_{1,t} \cup S_{2,t}} \frac{n_k}{m_t} w_t^k$
  if $t \leq 3$ **then**
    $w_{t+1} \leftarrow w_{1,t+1}$
ClientUpdate(k, w)**:**
$\beta \leftarrow$ split P_k into batches of size B
for *each local epoch* $i = 1$ ***to*** E **do**
  for *each batch* $b \in \beta$ **do**
    $w \leftarrow w - \eta \cdot \nabla \ell(w; b)$
return w

to the FedAvg process so that $S_{1,t}$, $S_{2,t}$, and S_t all participate in the aggregation process.

Regarding the issue of the server-side client not having the number of samples, we randomly select S_t clients out of the total K participants, and use their number of data samples to assign weights to the server-side client. After repeating this process over several rounds, the final GM is provided to the model requester.

With this training approach, even if malicious clients participate in the training process, the models CM_1 and CM_2 for each round t they obtain will both be affected by noise, meaning the accuracy of GM is not guaranteed. Therefore, this method provides a passive defense against model theft.

A. Accuracy-Noise Trade-off

Generally, as noise intensity (σ) increases, the average accuracy tends to decrease. but. there remains a very low probability that high accuracy will be maintained, and likewise, there is a low but non-zero probability of complete accuracy loss.

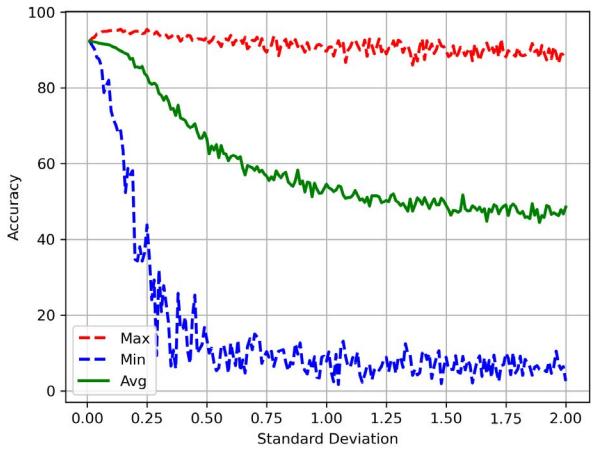


Fig. 2: Accuracy of CM_1 ($t = 10$)

Figure 2 shows a graph illustrating how the accuracy changes as the σ is increased for the model trained on the client-distributed model CM_1 at round 10. This graph presents the results of testing where the noise was regenerated 200 times for each noise intensity. The key observation is that the fluctuation in accuracy also increases as the σ rises. In other words, the variability of the accuracy increases, suggesting that the σ not only reduces accuracy but also raises the unpredictability of the results, affecting the consistency of the outcomes. These results point to two important conclusions. **First**, while the σ significantly affects accuracy, it does not necessarily guarantee the same level of performance degradation for the same intensity of noise. **Second**, as the σ increases, the range of accuracy degradation also becomes larger. This implies that when adjusting the σ, it is crucial to consider its impact on model performance thoroughly.

B. How to Control the Noise?

In each round, noise is added to the client-distributed models $CM_{1,t}$ and $CM_{2,t}$ as part of the federated learning process. Based on the concept of differential privacy, we can define ε -differentially private model.

Definition 2. (ε - differentially private model)
Let $\varepsilon > 0$ and $\xi \sim \mathcal{N}(0, \sigma^2)$ where $\xi \in \mathbb{R}^{C_1 \times H_1 \times W_1 \times K_1}$. Let $A(M, D)$ denote the accuracy of model M on dataset D. $CM + \xi$ is said to be ε-differentially private model if the following condition holds:

$$\Pr[A(GM, D) \in T] \leq e^{\varepsilon} \times \Pr[A(CM + \xi, D) \in T].$$

We can restrict the range of interest to specific intervals. For example, let $T = [60, 100]$, which corresponds to high-accuracy predictions. Here, the objective is ensuring that the global model GM_t maintains its accuracy within this range, while allowing degradation in the accuracy of client-distributed models $CM_{1,t}$ and $CM_{2,t}$. In this case, the probability of each client-distributed model's accuracy falling within this range must be smaller than that of the global model.

The privacy sensitivity parameter (ε) governs the intensity of the noise added to the model. As ε increases, the noise intensity σ grows, leading to greater privacy protection but also increasing variability in model predictions, especially within the range T. The trade-off between model privacy and model performance, particularly for high-accuracy predictions, is managed through this parameter.

While an infinite ε ensures maximal privacy protection, it results in diminishing model accuracy, and this balance must be carefully managed in practical applications.

We aim to calculate the probability that a model's output lies within a specific interval T using the *cumulative distribution function (CDF)*.

Let μ denote the mean and σ the standard deviation of a normal distribution. The Z-score for a value x is calculated as:

$$Z_x = \frac{x - \mu}{\sigma}.$$

We are interested in calculating the probability that a random variable X from this distribution lies within the interval $[\delta, 100]$. This probability is given by the difference between the CDF values at the Z-scores corresponding to 100 and δ:

$$P(\delta \leq X \leq 100) = P(Z_{100}) - P(Z_{\delta}).$$

Thus, the probability that the random variable X lies within the interval $[\delta, 100]$ can be expressed as:

$$P(\delta \leq X \leq 100) = \int_{-\infty}^{Z_{100}} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}\, dz - \int_{-\infty}^{Z_{\delta}} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}\, dz.$$

This represents the difference in the area under the standard normal curve between Z_{100} and Z_{δ}, which corresponds to the probability that X lies between δ and 100.

Suppose that model M is subjected to random noise on each parameter, following a normal distribution $\mathcal{N}(0, 2^2)$, while model M' is perturbed by noise following $\mathcal{N}(0, 0.1^2)$. As indicated earlier, model M may exhibit higher accuracy than model M', despite the increased noise intensity. This outcome is somewhat counterintuitive, as it suggests that a model with higher noise intensity can demonstrate greater accuracy. According to **Definition 2**, model M—which has a higher privacy sensitivity—also achieves higher accuracy, raising the question of whether lower-accuracy models are necessarily better suited as client-distributed models for protecting the global model. The answer is not straightforward.

Suppose client-distributed model CM's convolutional layer is A and global model GM's Convolutional Layer is B. Let $A^{\xi} = A + \xi$ where $\xi \sim \mathcal{N}(0, \sigma^2)$ and $\xi \in \mathbb{R}^{C_1 \times H_1 \times W_1 \times K_1}$. We can calculate Frobenius distance between A^{ξ} and B, which is based on the Frobenius norm. The Frobenius norm

Algorithm 2: `NoiseControlUnit`

Input:
ε: privacy sensitivity
n: number of trials
γ: increase in σ
Initialization:
$\sigma \leftarrow 0$ // Noise intensity value
(co,ex1,ex2) = ([ ], [ ], [ ]) // List of accuracy from n random tests on control group, CM_1 and CM_2
for $i = 1$ ***to*** n **do**
  co.append(Acc($CM_{1,t}$))
$\mu_{\text{co}} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \text{co}[i]$ //Compute Average
$\sigma_{\text{co}} \leftarrow \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\text{co}[i] - \mu_{\text{co}})^2}$
$Z_{100} \leftarrow \frac{100 - \mu_{\text{co}}}{\sigma_{\text{co}}}$ //Compute CDF
$Z_{\delta} \leftarrow \frac{\delta - \mu_{\text{co}}}{\sigma_{\text{co}}}$
$P_{co} \leftarrow P(Z_{100}) - P(Z_{\delta})$
while $\varepsilon_{test} \leq \varepsilon$ **do**
  $\sigma += \gamma$
  for $z = 1$ ***to*** n **do**
    $\xi \sim \mathcal{N}(0, \sigma^2), \quad \xi \in \mathbb{R}^{C_1 \times H_1 \times W_1 \times K_1}$
    $w_{1,t}^{\text{CONV1}} \leftarrow w_{1,t}^{\text{CONV1}} + \xi$
    $w_{2,t}^{\text{CONV1}} \leftarrow w_{2,t}^{\text{CONV1}} - \xi$
    $ex1$.append(Acc($CM_{1,t}$))
    $ex2$.append(Acc($CM_{2,t}$))
    $w_{1,t}^{\text{CONV1}} \leftarrow w_{1,t}^{\text{CONV1}} - \xi$
    $w_{2,t}^{\text{CONV1}} \leftarrow w_{2,t}^{\text{CONV1}} + \xi$
  $\mu_{ex1} = \frac{1}{n} \sum_{i=1}^{n} Acc_{1,t}[i]$ // Compute Averages
  $\mu_{ex2} = \frac{1}{n} \sum_{i=1}^{n} Acc_{2,t}[i]$
  $\sigma_{ex1} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (ex1[i] - \mu_{ex1})^2}$
  $\sigma_{ex2} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (ex2[i] - \mu_{ex2})^2}$
  $Z1_{100} \leftarrow \frac{100 - \mu_{\text{ex1}}}{\sigma_{\text{ex1}}}$ //Compute CDF $Z1_{\delta} \leftarrow \frac{\delta - \mu_{\text{ex1}}}{\sigma_{\text{ex1}}}$
  $Z2_{100} \leftarrow \frac{100 - \mu_{\text{ex2}}}{\sigma_{\text{ex2}}}$
  $Z2_{\delta} \leftarrow \frac{\delta - \mu_{\text{ex2}}}{\sigma_{\text{ex2}}}$
  $P_{ex1} \leftarrow P(Z1_{100}) - P(Z1_{\delta})$
  $P_{ex2} \leftarrow P(Z2_{100}) - P(Z2_{\delta})$
  $\varepsilon_{test} \leftarrow \min(\ln \frac{P_{co}}{P_{ex1}}, \ln \frac{P_{co}}{P_{ex2}})$
return σ

is advantageous in preserving the matrix structure.

$$||A^{\xi} - B||_F = \sqrt{\sum_{c,h,w,k}(A_{c,h,w,k} + \xi_{c,h,w,k} - B_{c,h,w,k})^2}. \tag{1}$$

Given that $\mathbb{E}[\xi_{c,h,w,k}] = 0$ and $\mathbb{E}[\xi^2_{c,h,w,k}] = \sigma^2$ (since $\xi_{c,h,w,k}$ is sampled from $\mathcal{N}(0, \sigma^2)$), we get:

$$\mathbb{E}[||A^{\xi} - B||_F^2] = \sum_{c,h,w,k}(A_{c,h,w,k} - B_{c,h,w,k})^2 + (C_1 \times H_1 \times W_1 \times K_1)\sigma^2. \tag{2}$$

By the definition of Expectation, the following relationship holds:

$$\sigma \propto \mathbb{E}[||A^{\xi} - B||_F^2] = \mathbb{E}[||A^{\xi} - B||_F]^2 + Var(A^{\xi} - B).$$

Here, the variance is always greater than or equal to 0. Therefore, if σ increases, the expectation of the distance between A^{ξ} and B increases. This indicates that the model with higher noise intensity will deviate more from the global model, potentially capturing feature maps that differ significantly from those learned by the global model. It suggests that the perturbed model is learning features that diverge substantially from those of the original global model. This can introduce significant discrepancies during model updates, potentially compromising the overall performance of the global model.

Therefore, it is insufficient to conclude that a model with higher accuracy is inherently less suitable for model privacy protection in a FL framework.

V. Experiments

A. Experiment Settings

For our experiments, we use a CNN model consists of two convolutional layers (32 and 64 filters, kernel size 3×3), followed by two dropout layers (probability of 0.25 and 0.5), and two fully connected layers (128 neurons and 10 output classes).

TABLE II: Hyperparameter values

Dataset	MNIST (IID)
Communication rounds	20 (Fig.3), 15 (Fig.4)
Learning rate η	0.0001
Batch size B	16
Local epochs E	10
Client participation	$S_{1,t} = 5$, $S_{2,t} = 5$ per round
Server-side clients S_t	20 (from 2nd round, FedAvg)
Privacy sensitivity ε	0.6
σ increase in size (γ)	0.01 (Fig. 5), 0.1 (Fig. 6–7)
Lower bound T ($\tau = \delta$)	60
Noise application	Conv1 parameters (from 4th round)
Optimizer	SGD

B. Results

In **Figure 3**, GM maintains high accuracy, similar to the baseline model (conventional FedAvg without global model protection), with minimal impact from the applied noise. Additionally, the accuracies of the client-distributed models are effectively restricted, showing that the model privacy protection mechanism is functioning as intended. Therefore,

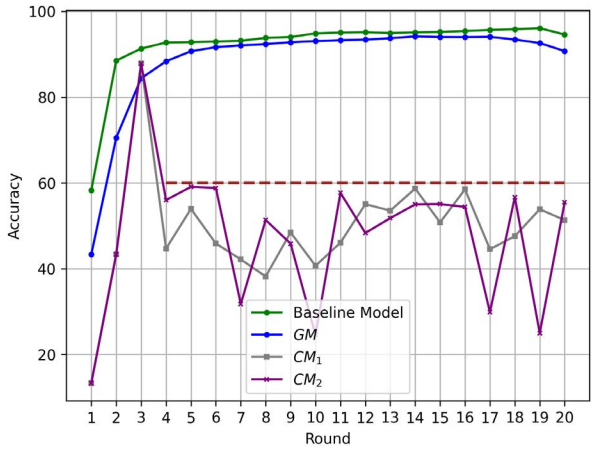


Fig. 3: Comparison of accuracy between the proposed model and the baseline model

the utility of the global model is kept at a similar level as in traditional FL, while the utilities of the client distributed models are kept below the predefined upper bound, making the model theft attack meaningless.

Figure 4-(a) presents a comparison of the model performance based on the number of server-side clients ($|S_t|$): 0, 20, and 40. The goal of server-side clients is to ensure stable model convergence by minimizing the accuracy degradation caused by noise variability. When too many server-side clients are applied, the convergence speed decreases, or the previous rounds are more likely to affect subsequent rounds. On the other hand, when no server-side clients are used, convergence can be faster, but the results are not as consistent.

Figure 4-(b) demonstrates the accuracy comparison across different proportions of $|S_1|$ and $|S_2|$, where the 1:1 proportion results in the best convergence for the global model. The experiment shows that a 1:1 ratio yields the best accuracy and convergence, supporting the effectiveness of evenly distributing clients across groups in **Algorithm 1**.

Figure 4-(c) illustrates the relationship between γ and σ. It shows that smaller γ (i.e., smaller increments in σ) leads to higher accuracy, but increases computational cost by the server.

VI. Conclusion

The method proposed in this study provides a robust mechanism for model privacy protection by introducing random noise into the global model parameters, effectively reducing the performance of client-distributed models. Experimental results demonstrate that the global model maintained performance comparable to the baseline, while successfully constraining the accuracy of the client-distributed models. This outcome supports the efficacy of the proposed model privacy protection mechanism. When compared to SOTA mechanisms (e.g., homomorphic encryption), the proposed method significantly reduces computational overhead on the client side, as, from the client's perspective, whole procedure does not

differ from traditional FL. This is a considerable advantage, as it avoids the computational burden typically associated with clients in standard FL applications while maintaining a high level of model privacy protection. However, the process of determining noise intensity and the noise map, which are based on model accuracy, introduces additional time complexity for the server. Specifically, the computation required by the **NoiseControlUnit** leads to delays that could pose challenges in real-time applications. This issue calls for the development of more efficient methods to adjust noise intensity, reducing latency while preserving model privacy protection. Although the proposed approach has been validated within a limited scope, specifically for certain datasets and model architectures, we expect that the approach will be applicable across a wide range of models and datasets because the process of adding noise to model parameters using our algorithm is quite generic and does not depend on the model architecture.

ACKNOWLEDGMENTS

This research was supported by MSIT Korea under NRF Korea (RS-2026-00557379, 60%). This research was also supported by the MSIT, Korea, under the Innovative Human Resource Development for Local Intellectualization support program (IITP-2026-RS-2022-00156360, 30%) and the Convergence Security Core Talent Training Business Support Program (IITP-2026-RS-2024-00426853, 10%) supervised by IITP.

REFERENCES

[1] B. McMahan *et al.*, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, vol. 54, PMLR, pp. 1273–1282, 2017.

[2] S. R. Pandey *et al.*, "A Crowdsourcing Framework for On-Device Federated Learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3241-3256, May 2020.

[3] B. Zhao *et al.*, "CrowdFL: Privacy-Preserving Mobile Crowdsensing System via Federated Learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 8, pp. 4607-4619, 2023.

[4] Y. Pan *et al.*, "FedSHE: Privacy Preserving and Efficient Federated Learning with Adaptive Segmented CKKS Homomorphic Encryption," *Cybersecurity*, vol. 7, no. 1, article 40, 2024.

[5] J. H. Cheon *et al.*, "Homomorphic Encryption for Arithmetic of Approximate Numbers," in *Adv. Cryptol. – ASIACRYPT*, vol. 23, Springer, pp. 409-437, 2017.

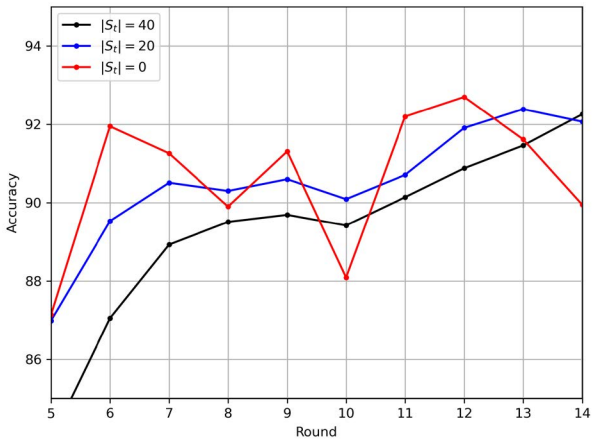
[6] J. Liang and R. Wang, "FedCIP: Federated Client Intellectual Property Protection with Traitor Tracking," *arXiv preprint arXiv:2306.01356*, 2023.

[7] S. Pal *et al.*, "ActiveThief: Model Extraction Using Active Learning and Unannotated Public Data," in *Proc. AAAI Conf. Artif. Intell.*, pp. 865-872, 2020.

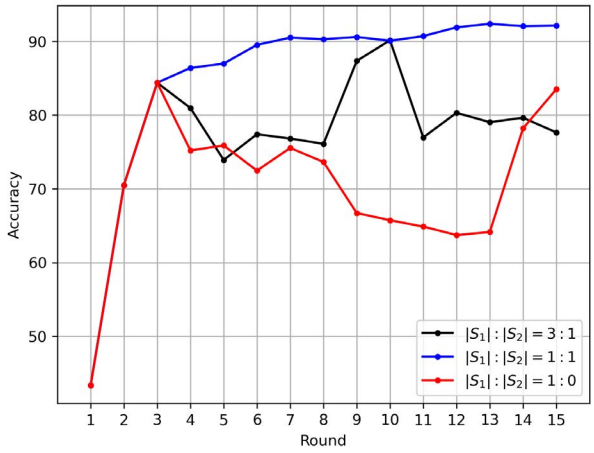
[8] K. Khaled *et al.*, "Efficient Defense Against Model Stealing Attacks on Convolutional Neural Networks," in *Proc. Int. Conf. Mach. Learn. Appl. (ICMLA)*, IEEE, pp. 45-52, 2023.

[9] C. Dwork, "Differential Privacy: A Survey of Results," in *Int. Conf. Theory Appl. Models Comput.*, Springer, pp. 1-19, 2008.

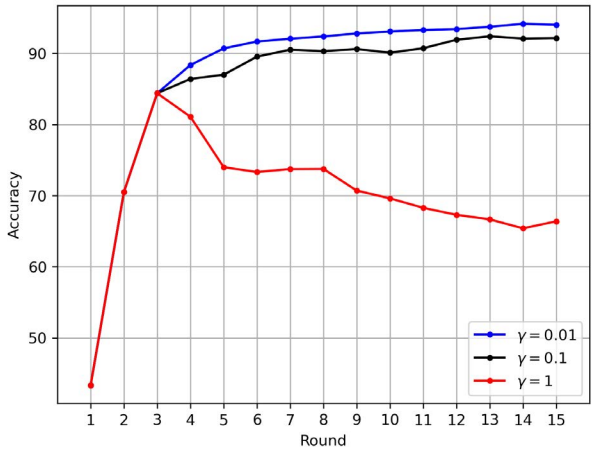
[10] M. Aitsam, "Differential Privacy Made Easy," in *Proc. Int. Conf. Emerg. Trends Electr., Control Telecommun. Eng. (ETECTE)*, IEEE, pp. 1-7, 2022.



(a) Impact of server-side clients



(b) Impact of $|S_1| : |S_2|$



(c) Impact of γ

Fig. 4: Comparison of experimental results across different settings