ARTIFACT EVALUATED USENIX AVAILABLE

ARTIFACT EVALUATED USENIX FUNCTIONAL

USENIX Security '26 Artifact Appendix: <United We Defend: Collaborative Membership Inference Defenses in Federated Learning>

Li Bai[1], Junxu Liu[1,2], Sen Zhang[1], Xinwei Zhang[1], Qingqing Ye[1], Haibo Hu[1,2] *
Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University[1]
PolyU Research Centre for Privacy and Security Technologies in Future Smart Systems[2]

A Artifact Appendix

A.1 Abstract

This artifact contains the core CoFedMID algorithm for defending against membership inference attacks in federated learning, along with detailed documentation for environment setup and demo execution. To facilitate understanding and reproducibility of our experiments, we provide an end-to-end demonstration that covers dataset preparation, federated learning training, and the implementation and evaluation of membership inference attacks, enabling readers to reproduce our results and extend our defense to new datasets or attack settings.

A.2 Description & Requirements

Our experiments were carried out on a machine running CUDA 12.5 with four NVIDIA GeForce RTX 3090 GPUs. The software environment is based on Python 3.9 and relied on several key libraries, including PyTorch, TorchVision, and Torchaudio. All required Python dependencies are listed in the `requirements.txt` file provided in our code repository.

A.2.1 Security, privacy, and ethical concerns

There are no security, privacy, or ethical concerns.

A.2.2 How to access

This artifact is accessible from the following URL: https://zenodo.org/records/17982729.

A.2.3 Hardware dependencies

We conducted our experiments on an Ubuntu system equipped with four NVIDIA GeForce RTX 3090 GPUs, each with 24 GB of memory. The artifact is expected to be compatible with other modern GPU devices.

*Corresponding author: haibo.hu@polyu.edu.hk

A.2.4 Software dependencies

The software environment is based on Python 3.9.25 and relies on the following key libraries: PyTorch 2.6.0, TorchVision 0.21.0, Torchaudio 2.6.0, and Opacus 1.5.4. These specific versions were used in all experiments and are recommended to ensure full compatibility and reproducibility. Additional standard Python dependencies are listed in detail in the `requirements.txt` file.

A.2.5 Benchmarks

Four benchmark datasets are used in our experiments: CIFAR10, CIFAR100, and TinyImageNet. The former two are readily available through PyTorch's built-in dataset utilities. For TinyImageNet, detailed instructions for downloading and preprocessing are provided in the `README.md`. Besides, we focus on two families of image classifier models: ResNet, and WideResNet.

A.3 Set-up

We provide some shell scripts to automate the evaluation.

A.3.1 Installation

We provide a `requirements.txt` file to facilitate environment reproduction and script execution. The necessary dependencies can be installed by running the following commands:

```
conda create -n cofedmid python=3.9 -y
conda activate cofedmid
pip install -r requirements.txt
```

An additional script, `run_verify.sh`, is also provided to test the installation environment.

A.3.2 Basic Test

Basic tests can be run by:

```
bash run_cofedmid.sh
```

For more detailed and comprehensive evaluations and demonstrations, please refer to the `README.md` file for additional information.

A.4 Evaluation workflow

Our experiments mainly rely on two scripts, namely `train_FL.py` and `mia_attack.py`. The former is used to train a federated learning system, while the latter is designed to conduct membership inference attacks in the federated learning setting. Our defense mechanisms are implemented in `train_FL.py` by enabling specific defense options and configuring corresponding hyperparameters. Finally, the training and attack results are saved in CSV format for further analysis.

A.4.1 Major Claims

(C1): CoFedMID achieves utility performance comparable to that of undefended models, as supported by the results of E1 and E2. Further analysis is provided in Section 5.2.

(C2): CoFedMID effectively mitigates membership inference risks in federated learning, as demonstrated by the attack results in E3. For instance, the results reported in Tables 1 and 2 for CIFAR100 with ResNet18 illustrate this effect.

A.4.2 Experiments

We provide a detailed demonstration for reproducing our experiments using CIFAR100 and ResNet18 as an example. Experiments on other datasets and model architectures can be conducted by modifying the corresponding hyperparameters, as described in Section 5.1 of our paper.

(E1): [Undefended Models] We train undefended federated learning models using `train_FL.py`.

How to: Download the required datasets and configure the federated learning setup.

Preparation: For TinyImageNet, run the preprocessing script `utils/preprocess_tiny_imagenet.py`. Other datasets do not require preprocessing.

Execution: Execute `train_FL.py` with the hyperparameters `--defense none` and `--shuffle none`. A demo script `run_baseline.sh` is also provided in our code repositories for convenience.

Results: Collect the training results, which include the training configurations and the achieved test accuracy. For instance, the test accuracy of 0.4669 can be found in `info_cifar100_resnet18.csv`.

(E2): [CoFedMID Implementation] We apply CoFedMID to standard federated learning.

How to: We implement CoFedMID by configuring the federated learning setup to enable CoFedMID defense strategies. The main hyperparameters of CoFedMID include three settings: the number of trusted users (`--trusted_users T`) and the defense strategy (`--defense clsplit-mab-hamp` and `--shuffle layadd`). The former, `--defense clsplit-mab-hamp`, activates the *Class-guided Partition* and *Utility-aware Compensation* components, while the latter, `--shuffle layadd`, enables the *Aggregation-neutral Perturbation* component.

Preparation: Same as E1.

Execution: A demo script `run_cofedmid.sh` is provided for running CoFedMID with all modules enabled under the Pair and Half scenarios.

Results: Similar to E1, the training results are saved in CSV files. For example, the test accuracy of 0.4509 can be found in `info_cifar100_resnet18.csv` for scenarios where a pair of clients collaborate.

(E3): [Attack Simulation] We perform state-of-the-art membership inference attacks in federated learning to evaluate the effectiveness of defense strategies.

How to: Using the intermediate results of local and global models, we run `mia_attack.py` to simulate potential attacks.

Preparation: During E1 or E2, intermediate results were already saved and serve as the attack dataset.

Execution: The attack commands are provided in `run_baseline.sh` and `run_cofedmid.sh` to execute the attacks. Example usage:

```
python mia_attack.py
--dataset $dataset  --defense $defense
--shuffle $shuffle  --folder $folder
```

Results: The attack results are saved in CSV files, such as "res_*.csv", including metrics such as MIA, AUC, and TPR@FPR=0.1/0.01/0.001. Results corresponding to CoFedMID can be identified by filtering rows where `defense==clsplit-mab-hamp-layadd`.

In fact, our repository includes detailed demo scripts described in `README.md`, covering training, attack scripts, and result visualization using Jupyter notebooks.

A.5 Notes on Reusability

CoFedMID is designed in a modular manner and can be extended to defend against other privacy attacks in federated learning, such as gradient inversion attacks. In particular, the *Aggregation-neutral Perturbation* module is reusable and can perturb client gradients while preserving the correctness of the global aggregation process.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.