Fan Mo and Hamed Haddadi *Imperial College London, UK*
Kleomenis Katevas, Eduard Marin, Diego Perino and Nicolas Kourtellis *Telefonica Research, Barcelona, Spain*

Editors: Nicholas D. Lane and Xia Zhou

PPFL: ENHANCING PRIVACY IN FEDERATED LEARNING WITH CONFIDENTIAL COMPUTING

Excerpted from "PPFL: privacy-preserving federated learning with trusted execution environments" from MobiSys '21: *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* with permission. https://dl.acm.org/doi/10.1145/3458864.3466628 ©ACM 2021

Mobile networks and devices provide the users with ubiquitous connectivity, while many of their functionality and business models rely on data analysis and processing. In this context, Machine Learning (ML) plays a key role and has been successfully leveraged by the different actors in the mobile ecosystem (e.g., application and Operating System developers, vendors, network operators, etc.). Traditional ML designs assume (user) data are collected and models are trained in a centralized location. However, this approach has privacy consequences related to data collection and processing. Such concerns have incentivized the scientific community to design and develop Privacy-preserving ML methods, including techniques like Federated Learning (FL) where the ML model is trained or personalized on user devices close to the data; Differential Privacy, where data are manipulated to limit the disclosure of private information; Trusted Execution Environments (TEE), where most of the computation is run under a secure/private environment; and Multi-Party Computation, a cryptographic technique that allows various parties to run joint computations without revealing their private data to each other.

Federated Learning

FL has received a lot of attention and is being deployed in real-world systems such as the Google Keyboard, and even envisioned to be a future "as-a-Service" solution [3]. The main idea of FL is to train deep neural networks (DNNs) locally on multiple mobile devices and build an aggregated global model on a server. This allows training a model without the need for mobile users to reveal their data, thus preserving their privacy. As illustrated in Figure 1, the model is built in an iterative fashion, in FL rounds. In each round, the server selects a subset of devices, and sends them the current global model. Each device updates this model by training it using its local data, and sends its local model version (i.e., the updated model parameters/gradients) to the server, which then aggregates all these updates to generate a new global model for the next round.

Can Federated Learning guarantee privacy during ML training?

Although user data are not collected at a centralized location during FL, recent works have shown that adversaries can execute various types of privacy attacks to retrieve sensitive information from the FL model parameters themselves, thus breaking the initial privacy promises behind FL. Prominent examples of such attacks are data reconstruction [14] and various types of inference attacks [6,10]. Fundamentally, these attacks are possible because as models learn to achieve their main task, they also learn irrelevant information from users' training data that is inadvertently embedded in the final model [7,13].

Data Reconstruction Attacks aim at reconstructing original input data based on the observed model or its gradients. These attacks invert model gradients based on generative adversarial attack-similar techniques [14] and consequently reconstruct the corresponding original data used to produce the gradients. As the server typically observes updated models of each client in plaintext, it is more likely for this type of leakage to exist at the server.

Property Inference Attacks aim at inferring the value of private properties in the training data. These attacks are achieved by building a binary classifier trained on model gradients updated with auxiliary data. Even though clients in FL only observe multiple snapshots

Illustration, istockphoto.com

of broadcast global models that have been linearly aggregated on participating clients' updates, property information can still be well preserved, providing attack points to client-side adversaries.

Membership Inference Attacks aim at learning whether specific data instances are present in the training dataset. Such attacks can be performed by building a binary classifier [10] similar to property inference attacks. Membership is *high-level* latent information and the risk can exist on both server and client sides. Adversaries can even perform this attack on the final (well-trained) model and its last layer [10,13].

State-of-the-art countermeasures and limitations

Existing solutions to prevent the above attacks can be grouped into three main categories depending on whether they rely on: (i) homomorphic encryption, (ii) multi-party computation, or (iii) differential privacy. While homomorphic encryption is practical in both high-end and mobile devices, for now, it only supports a limited number of arithmetic operations in the encrypted domain. Alternatively, fully homomorphic encryption allows arbitrary operations in the encrypted domain, thus supporting ML, but at the same time, it introduces a high computational overhead, making it impractical for mobile devices. Similarly, using secure multi-party computation has significant computational overhead. In addition, in some cases, differential privacy can fail to provide sufficient privacy, and it can negatively impact the utility and fairness of the model, as well as system performance.

More recently, the use of hardware-based TEEs has been proposed as a promising way to preclude privacy attacks against ML models. TEEs allow for data to be stored securely and execute arbitrary code on an untrusted device almost at native speed through secure memory compartments. Such advantages – together with the recent commoditization of TEEs both in high-end and mobile devices – make TEEs an ideal candidate to achieve full, privacy-preserving ML training. However, in order to keep the Trusted Computing Base (TCB) as small as possible, current TEEs have limited memory. This makes it impossible to simultaneously place all DNN layers inside the TEE. As a result, prior work [9] has opted for using TEEs to conceal only the most sensitive DNN layers from adversaries, leaving the rest of the layers unprotected. While this approach was sufficient to alleviate some attacks against traditional ML, where clients obtain only the final model, the attack surface of FL scenarios is significantly larger. FL client devices can observe distinct snapshots of the model throughout the training, allowing them to realize attacks at different stages.

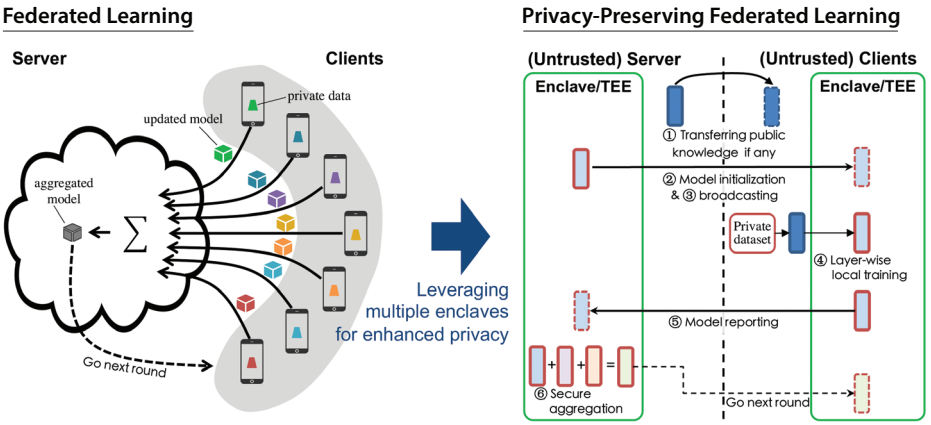

FIGURE 1. A schematic diagram of PPFL framework, when executing FL training in a greedy, layer-wise fashion.

PPFL: PRIVACY-PRESERVING FEDERATED LEARNING

PPFL is the first practical framework to fully prevent private information leakage at both server and client-side under FL scenarios. We consider two types of (passive) adversaries: (i) users of client devices with access to distinct snapshots of the global model and (ii) the server's owner (e.g., a cloud or edge provider) who has access to the updated model gradients. Adversaries are assumed to be *honest-but-curious*, meaning they allow FL algorithms to run as intended, while trying to infer as much information as possible from the global model or gradients. Adversaries can have full control (i.e., root privileges) of the server or a client device and can perform their attacks against any DNN layer.

PPFL is based on greedy layer-wise training and aggregation, overcoming the constraints posed by the limited TEE memory, and providing comparable accuracy of complete model training at the price of a tolerable delay. In PPFL, lower-level information (i.e., original data and attributes) is not exposed because updated gradients during training are not accessible from adversaries (they happen inside the TEEs). This protects against data reconstruction and property inference attacks. However, when one of such layers is exposed after convergence, there is a risk of membership inference attacks. We follow a more practical approach based on the observation that membership-related information is only sensitive in the last DNN layer, making it vulnerable as indicated in previous research [9–11]. To avoid this risk on the final model, PPFL can keep the last layer inside the clients' TEEs after training. Finally, our layer-wise approach supports sophisticated settings such as training one or more layers (as a block) in each iteration, which can potentially better deal with heterogeneous data at the client-side and speed up the training process.

Design of PPFL system

Transferring Knowledge if any (Step 1): In cases of typical ML, tasks such as image recognition where public knowledge is available in the form of pre-trained models, the server can transfer this knowledge to bootstrap and speed up the training process. This knowledge is usually contained in the first layers of a model. Thus, the clients leave the first layers frozen and only train the last several layers of the global model.

Model Initialization and Broadcasting (Steps 2 & 3): The server configures the model architecture, decides the layers to be protected by TEEs, and then initializes model parameters inside the TEE. In addition, the server configures other training hyper-parameters, such as learning rate, batch size, etc., before transmitting these settings to clients.

Layer-wise Local Training (Step 4): After model transmission and configuration using secure channels, each client starts local training on their data on each layer via a model partitioned execution technique, which conducts model training (including both forward and backward passes) across Rich OS and TEEs.

Reporting and Aggregation (Steps 5 & 6): Once local training of a layer is completed inside TEEs, all participating clients report the layer parameters to the server through secure channels. Finally, the server securely aggregates the received parameters within its TEE and applies aggregation, resulting in a new global model layer.

Evaluation of PPFL System

Prototype: We implement the PPFL client-side by building on top of DarkneTZ [9], to support on-device FL with Arm TrustZone, running on a HiKey 960 Board (16MiB TrustZone TEE secure memory). We implement the PPFL server-side on Microsoft Open-Enclave with Intel SGX and run it on an Intel Next Unit of Computing (i3-8109U CPU) with SGX-enabled capabilities. We developed a set of bash shell scripts to control the FL process and create the communication channels.

Model and Datasets: We focus on Convolutional Neural Networks, since the privacy risks we consider have been extensively studied on such DNNs [6,10]. Here, we only show evaluation results of the AlexNet model trained on the CIFAR10 dataset. More results about our tested models and datasets can be found in [8]. The dataset is split in two ways: (i) Independent and Identically Distributed (IID); and (ii) Non-IID.

PPFL Robustness to Privacy Attacks: As shown in Table 1, data reconstruction attacks on PPFL can only reconstruct a fully noised image for any target image (i.e., an MSE of ~1.3 for the specific dataset), while property inference attacks on PPFL always report a random guess on private properties (i.e., an AUC of ~0.5). Membership inference attacks' advantage on PPFL is significantly dropped (i.e., Precision≈0.5).

PPFL Utility and Communication Cost: PPFL can achieve comparable ML utility with end-to-end (regular) FL. While layer-wise FL increases the total number of communication rounds needed to finish all layers, it can reach the same test accuracy as end-to-end FL with fewer rounds (0.538x) and amount of communication in bytes (1.002x). Instead, during the early stage of layer-wise training, it can already reach good ML performance, and in some cases even better than training the entire model at once. Consequently, as shown in Figure 2, due to the needed rounds being fewer, the amount of communication can also be reduced. Besides, we find that training 2-layer blocks decreases communication cost by at least half (check [8] for more details).

PPFL System Performance: Most system cost comes from clients' local training in the whole system: up to ~15% CPU time, ~18% memory usage, and ~21% energy consumption in client cost when training different models and data, compared to training without TEEs, and slightly increased system overhead (i.e., CPU time, memory usage, energy consumption) in cases of small models.

TABLE 1. Results of three privacy-related attacks on PPFL vs. end-to-end FL. Average score reported with 95% confidence interval in parenthesis.

Learning Method	Privacy-related Attacks		
	Data Reconstruction (MSE scores)	Property Inference (AUC scores)	Membership Inference (Precision scores)
End to End FL	 0.017 (0.01)	0.930 (0.03)	0.874 (0.01)
PPFL	 ~1.3	~0.5	0.506 (0.01)

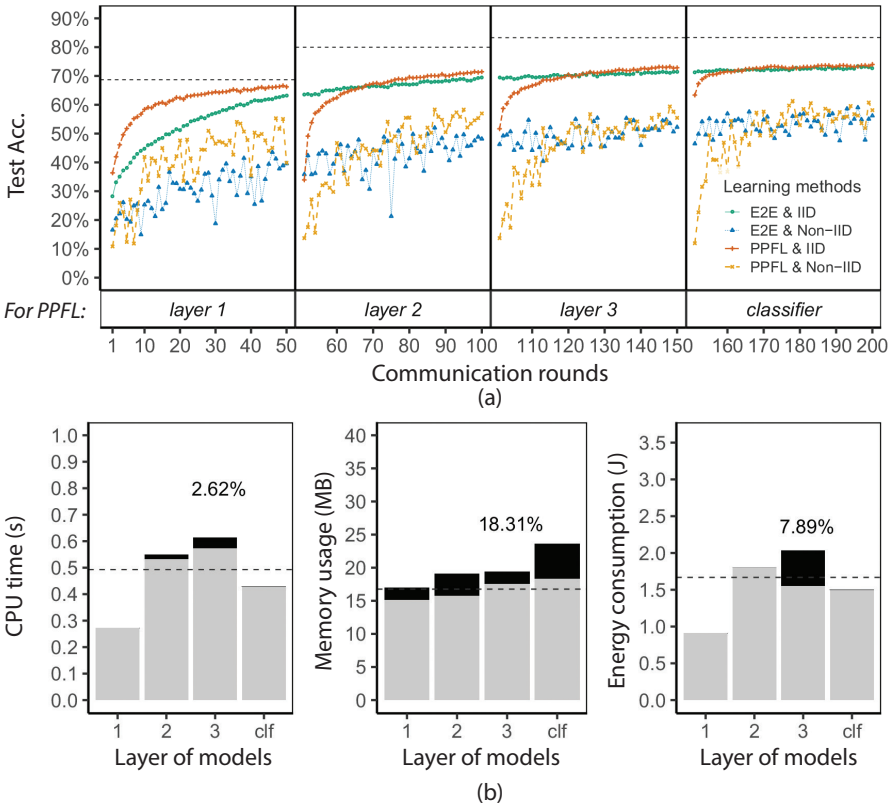

FIGURE 2. (a) Test accuracy of PPFL and End-to-end (E2E) FL. Horizontal dashed lines refer to the accuracy that E2E FL reaches after 50 communication rounds. (b) System performance of PPFL. Light grey bars refer to learning without TEEs, and black bars refer to overhead using TEEs. Percentage overhead (%) is shown above these bars. Horizontal dashed lines refer to E2E FL cost.

OPEN PROBLEMS AND CHALLENGES

Active Dishonest Attacks

The attacks tested here assume the classic "honest-but-curious" adversary. In FL, however, there are also dishonest attacks such as backdoor or poisoning attacks [1], with the goal of actively changing the global model

behavior, e.g., for surreptitious unauthorized access to the global model. How TEEs' security properties can defend against such attacks can be a potential research question.

Privacy and Cost Trade-off

PPFL guarantees "full" privacy by keeping layers inside TEEs. However, executing computations in secure environments inevitably leads to system costs. To reduce such costs, one can relax their privacy requirements, potentially increasing privacy risks of inference attacks with a higher "advantage." We also expect that by dropping clients already achieving good performance when training latter layers, we could gain better performance. This may further benefit personalization and achieve better privacy, utility, and cost trade-offs.

Accelerating Local Training

PPFL only uses the client device's CPU for local training. Training each layer does not introduce parallel processing on a device. Indeed, more effective ways to perform this compute load can be devised. One way is that clients could use specialized processors (i.e., GPUs) to accelerate training. However, as GPU-TEE still requires small TCB to restrict attack surface, PPFL design can provide a way to leverage limited TEE space for privacy-preserving local training.

Federated Learning Paradigms

PPFL was tested with FedAvg [5], but there are other compatible state-of-art FL paradigms. PPFL leverages greedy layer-wise learning but does not modify the hyper-parameter determination and loss function (already improved in FedProx [4]) or aggregation (which is neuron matching-based in FedMA [12]). Besides, PPFL is compatible with other privacy-preserving techniques (e.g., differential privacy) or Policy-based FL [2]. This is useful during the model usage phase, where some users may not have TEEs. ■

Fan Mo is a Ph.D. candidate of Design Engineering at Systems and Algorithms Laboratory, Imperial College London, U.K. His research focuses on data privacy and trustworthiness in machine learning at the edge, machine learning in confidential computing, and distributed machine learning on lightweight mobile/IoT devices.

Hamed Haddadi is a Reader in Human-Centred Systems at Imperial College London. In his industrial role, he is a visiting professor and the chief scientist at Brave Software, where he works on developing privacy-preserving analytics protocols. He enjoys designing and building systems that enable better use of our digital footprint, while respecting users' privacy.

Kleomenis Katevas is a research scientist at Telefonica Research in Barcelona, Spain. His research focuses on areas of mobile systems, privacy-preserving machine learning and human-computer interaction. He received his Ph.D. in computer science from Queen Mary University of London, UK.

Eduard Marin is a research scientist at Telefonica Research, in Barcelona, Spain. His research interests fall in the intersection between security and networks. He received his Ph.D. from the COSIC research group (KU Leuven, Belgium), then became a visiting research fellow in the SPRITZ group at the University of Padua (Italy) and a research fellow in the Security and Privacy group at the University of Birmingham, UK.

Diego Perino is the director of Telefonica Research, in Barcelona, Spain. Prior to Telefonica, he worked at Bell Labs, NICTA, Orange Labs. He received his Ph.D. in Computer Science from the Paris Diderot-Paris 7, M.S. in Networking engineering at Politecnico di Torino and Eurecom Institute of Sophia Antipolis.

Nicolas Kourtellis is a senior research scientist at Telefonica Research, in Barcelona, Spain. He currently focuses on studying challenges in cyber-security and privacy, web transparency, online user tracking and privacy-preserving machine learning methods. He holds a Ph.D. in Computer Science and Engineering from the University of South Florida, USA.

REFERENCES

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, PMLR, 2938–2948. https://proceedings.mlr.press/v108/bagdasaryan20a.html

[2] Kleomenis Katevas, Eugene Bagdasaryan, Jason Waterman, Mohamad Mounir Safadieh, Eleanor Birrell, Hamed Haddadi, and Deborah Estrin. 2021. Policy-based federated learning. http://arxiv.org/abs/2003.06612

[3] Nicolas Kourtellis, Kleomenis Katevas, and Diego Perino. 2020. FLaaS: Federated learning as a service. *Proceedings of the 1st Workshop on Distributed Machine Learning* (DistributedML'20), ACM, New York, NY, USA, 7–13. DOI:https://doi.org/10.1145/3426745.3431337

[4] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems 2*, 429–450.

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, PMLR, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[6] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. *2019 IEEE Symposium on Security and Privacy (SP)*, 691–706. https://doi.org/10.1109/SP.2019.00029

[7] Fan Mo, Anastasia Borovykh, Mohammad Malekzadeh, Hamed Haddadi, and Soteris Demetriou. 2020. Layer-wise characterization of latent information leakage in federated learning. *International Conference of Learning Representations: Workshop on Distributed and Private Machine Learning (DPML).*

[8] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. 2021. PPFL: Privacy-preserving federated learning with trusted execution environments. *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (MobiSys '21), ACM, New York, NY, USA, 94–108. https://doi.org/10.1145/3458864.3466628

[9] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. 2020. DarkneTZ: Towards model privacy at the edge using trusted execution environments. *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services* (June 2020), 161–174. https://doi.org/10.1145/3386901.3388946

[10] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. 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), 739–753. https://doi.org/10.1109/SP.2019.00065

[11] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Herve Jegou. 2019. White-box vs black-box: Bayes optimal strategies for membership inference. *Proceedings of the 36th International Conference on Machine Learning*, PMLR, 5558–5567. https://proceedings.mlr.press/v97/sablayrolles19a.html

[12] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2019. Federated learning with matched averaging. *International Conference on Learning Representations.* https://openreview.net/forum?id=BkluqlSFDS

[13] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 268–282. https://doi.org/10.1109/CSF.2018.00027

[14] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in Neural Information Processing Systems*, Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/hash/60a6c4002cc7b29142def8871531281a-Abstract.html