MixNN: Protection of Federated Learning Against Inference Attacks by Mixing Neural Network Layers

Thomas Lebrun
thomas.lebrun@inria.fr
Univ Lyon, INSA Lyon, Inria, CITI
Lyon, France

Antoine Boutet
antoine.boutet@insa-lyon.fr
Univ Lyon, INSA Lyon, Inria, CITI
Lyon, France

Jan Aalmoes
jan.aalmoes@inria.fr
Univ Lyon, INSA Lyon, Inria, CITI
Lyon, France

Adrien Baud
adrien.baud@inria.fr
Univ Lyon, INSA Lyon, Inria, CITI
Lyon, France

ABSTRACT

Machine Learning (ML) has emerged as a core technology to provide learning models to perform complex tasks. Boosted by Machine Learning as a Service (MLaaS), the number of applications relying on ML capabilities is ever increasing. However, ML models are the source of different privacy violations through passive or active attacks from different entities. In this paper, we present MixNN a proxy-based privacy-preserving system for federated learning to protect the privacy of participants against a curious or malicious aggregation server trying to infer sensitive information (i.e., membership and attribute inferences). MixNN receives the model updates from participants and mixes layers between participants before sending the mixed updates to the aggregation server. This mixing strategy drastically reduces privacy leaks without any trade-off with utility. Indeed, mixing the updates of the model has no impact on the result of the aggregation of the updates computed by the server. We report on an extensive evaluation of MixNN using several datasets and neural networks architectures to quantify privacy leakage through membership and attribute inference attacks as well the robustness of the protection. We show that MixNN significantly limits both the membership and attribute inferences compared to a baseline using model compression and noisy gradient (well known to damage the utility) while keeping the same level of utility as classic federated learning.

KEYWORDS

Privacy, Machine Learning, Federated Learning, Inference Attacks

1 INTRODUCTION

The collection of personal data is a subject firmly grounded in public debates. The growing awareness of the population on privacy issues led to stronger regulations on data protection (e.g., GDPR, HIPAA) and contributed to the appearance of new services making privacy an incentive vector such as privacy-based search engine (e.g., Duckduckgo, Qwant), web browsing (e.g., Web Proxy, Tor, Brave), or mailing (e.g., Protonmail). These services rely on infrastructures setup by companies, nonprofit organizations promoting privacy, or are fully peer-to-peer involving devices of end-users.

However, personal and private data is still the fuel of all desires. In this context, Machine Learning (ML) has emerged as a core technology to analyze and provide learning models from large volumes of data and to perform complex tasks such as classifications, predictions or clustering. The success of ML has driven different providers to launch Machine Learning as a Service (MLaaS) engines to make ML operation easier for anyone, without the cost and time to build in-house infrastructures. These new services has led to an ever increasing number of new applications or services relying on ML capabilities in different domains such as computer vision, health analytic and speech recognition to name a few. However, it has been showed that ML models may leak information in the training data [18, 39, 48]. The fact that many applications using this technology involve the collection and processing of personal and sensitive data has raised privacy concerns [16].

Despite being popular, the memorization of training data by a ML model is the source of different privacy violations such as membership, property and attribute inference through passive or active attacks. Membership inference [24, 32] refers to the capacity of an adversary to identify if a data point (or the data of an individual) has been used to train the target model. This attack has a serious privacy implication if the model is training with sensitive information (e.g., data from people with certain health status). Property inference [19, 47], in turn, corresponds to the inference by an adversary of the properties of training data such as the features that characterize each class. This property inference can also concern a subset of the training inputs. This ability to learn from training data is desired if the inference is directly related to the main task of the model. By contrast, attribute inference [29] corresponds to the fact that an adversary is able to infer an unintended and undesired attribute not correlated to class's characteristic feature. Root causes related to these attack surfaces as well as the link between utility (e.g., through model overfitting [38, 45]) and privacy are not well understood.

Recently, Federated Learning (FL) [7] has emerged as promising privacy-by-design alternatives to decentralized learning schemes. In such a collaborative scheme, personal data never leaves the user device. Instead, devices (computing and refining a learning model with their own data) and a central server (aggregating models) work together to build a global learning model. This new ML scheme has attracted many attention these last years, not only from the research community but also from major Internet companies, suggesting future deployments. For instance, Google already exploits FL for next-word prediction in a virtual keyboard for smartphones [21]. While the FL scheme is a clear step forward towards enforcing

users' privacy, it still suffers from a large ML-based attack surface including membership, property and attribute inference from participants or from the server. Different protection mechanisms to limit inference capabilities of an adversary have been proposed [31]. For instance, some solutions [42, 46] are based on perturbation in order to reveal only a noisy information to the server, such as differential privacy. However, these solutions significantly damage the accuracy of the model and its capacity to converge [41]. Secure aggregation relying on a cryptographic scheme has been also proposed [6, 8, 9]. Similar to MixNN, this solution ensures that the server is only aware of the aggregate of all models, keeping the model of each participant (and the associated inference) private. Although the overhead of this solution remains low, the underlying cryptographic scheme requires the participation of the server in the protection. We argue that such solutions are not deployed in practice. Indeed, few companies accept to afford the additional cost of the protection. For instance, Private Information Retrieval (PIR) protocols which follow similar cryptographic scheme to protect the profile of users are not widely adopted in practice. Moreover, a curious or malicious server trying to infer information from participants will certainly not adopt such a protection.

In this paper, we present MixNN, a new privacy-preserving service for FL against inference attacks from a curious aggregation server. To achieve that, MixNN relies on a proxy mixing the layers of the model updates (also named parameter updates) among participants before sending them to the aggregate server. Like Mixnets to ensure anonymity in information routing [12], mixing the layers of the participants' updates of neural network prevents inference attacks (i.e., both membership and attribute inference attacks) without decreasing the accuracy of the aggregated model. This solution, albeit simple, leads to drastically improving the privacy without any trade-off with utility. In addition, MixNN is transparent to the FL service, participants only need to configure a web proxy for the associated traffic. To make the deployment of MixNN easier by anyone (e.g., operate by an individual or non profit organizations willing to protect privacy) and possibly on an untrusted infrastructure, the proxy mixing the neural network layers is running inside an SGX enclave ensuring confidentiality and attestation on its behavior. Interestingly enough, the behavior of the proxy can be adapted according to the expected security and privacy guarantees. For instance, the proxy can aggregate itself the model updates or can adopt another aggregation scheme to improve the robustness against model poisoning or backdoor attacks [5] by replacing averaging with robust estimators such as geometric median. In this case, the utility-privacy-performance trade-off can change.

To illustrate the capability of MixNN to protect privacy while maintaining the same level of utility, we implemented MixNN and experimentally evaluated it with several datasets and neural network architectures. We also implemented both membership and attribute inference attacks to quantity and compare privacy leakage of MixNN against classical federated learning scheme, a model compression scheme, and a baseline using perturbation (i.e., noise) to protect the model updates (widely used in Differential Privacy). We show that MixNN drastically reduces the membership inference compared to other baselines (on average up to 73.9%, 73.8%, and -0.2% less inference against a classical FL, model compression and LDP, respectively), and limits the attribute inference (on average up to 13.8%, 14.6%, and 12.9% less inference against a classical FL, model compression and LDP, respectively) without decreasing the accuracy of the global aggregated model. Moreover, we show that reconstructing the update of a participant (by identifying the layers of an individual among the mixed updates) is costly and a difficult task which gives poor accuracy. Finally, we show that the MixNN proxy is scalable and introduces only a small latency on the model updates. The source code of MixNN as well as the experiments and datasets are publicly available [1].

The remainder of this paper is organized as follows. Section 2 presents background, Section 3 defines the problem and the threat model, Section 4 explains the design and the implementation of MixNN, Section 5 presents our evaluation setup, Section 6 reports the evaluation of MixNN, Section 7 reviews related work, and Section 8 concludes this paper.

2 BACKGROUND

In this section, we review background related to Neural Networks 2.1, Federated Learning 2.2, Inference Attacks 2.3, Mixnet 2.4, and Intel SGX 2.5.

2.1 Neural Networks

An ML model is a function $f(\theta) : X \mapsto Y$ parameterized by a set of parameters θ, where X denotes the input (or feature) space ($X = x_1, x_2, ..., x_k$), and Y the output space ($Y = y_1, y_2$). Training an ML model corresponds to find the optimal set of parameters θ that fits the training data. This is done by optimizing an objective function (loss) which penalizes the model when it is wrong. For instance, if we consider a classification task trained through a supervised learning, parameters θ are updated if the model misclassifies training data.

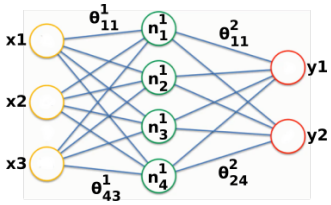


Figure 1: Example of Neural Network.

Neural networks are a family of ML models which have become popular for a variety of ML tasks. A neural network is composed of multiple layers of non-linear mappings from input to intermediate hidden states (or hidden layers) and then to output where each layer transforms the output of the preceding layer to produce input for the next layer. The topology of the connections between layers and the type of considered transformation function are task-dependent and impact the accuracy of the model. For instance, convolutional layers account for locality where each neuron receives a restricted input space while a neuron receives the entire previous layer in a fully connected layer.

[1] https://gitlab.inria.fr/abaud/mixnn

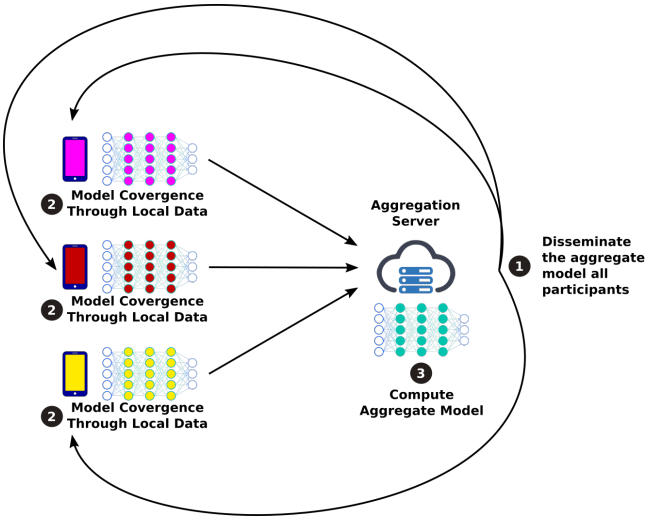


Figure 2: Operating flow of Federated Learning.

A neural network f is composed of a collection of n hidden layers ($f = (l_1, l_2, ..., l_n)$). Each layer l_i is composed of a set of m neurons ($l_i = (n_1^i, n_2^i, ...n_m^i)$). For input X, the output of the neural network, can be formally written as:

$$f(\theta) = F_n(F_{n-1}(...F_2(F_1(X)))),$$

where X is the input, n the number of layers, F_i represents a transformation function of the layer l_i, and θ is the set of floating-point weights associated with each connection between two neurons of different layers. Considering a fully connected neural network, (as depicted Figure 1), θ_{ab}^t represents the weight connecting the node n_a^t to the node n_b^{t-1}. These weights are updated during training according to the method to optimize the objective function. In this work, we consider Stochastic Gradient Descent (SGD) to do this optimization. SGD is an iterative approach where the optimizer receives a batch of training data and updates the model parameters θ at each iteration according to both the direction of the gradient of the objective function and a learning rate η which scales the update. Once the gradient is close to zero, the model has converged to a local minimum and the training is finished. The model is evaluated through its accuracy over testing data points not used to train the model. The hyperparameters refer to the set of tunable parameters not related to the neural network (e.g., weights associated to connection) such as the number of training iterations, the size of the training batch or the learning rate.

2.2 Federated Learning

Federated Learning (FL) is a collaborative learning scheme to train an ML model [7]. In such a scheme, personal data never leaves the device of participants. Instead, devices train a ML model locally and interact with a central server to build a global learning model.

The iterative-based operating flow of classical FL is depicted Figure 2. Each iteration contains three steps. First, the aggregation server disseminates a global model to participants (step ❶ in the figure). Each participant then trains and refines this model with its own data stored locally (step ❷). After this local training, each participant holds its own variation of the model sent by the server. Participants then send their updated model parameters to the aggregation server. Finally, the server aggregates all these updates to generate a new global model (step ❸) which will be disseminated to participants in the next iteration. Iteratively, the global model maintained on the server converges without requiring access to the personal data of participants.

2.3 Inference Attacks

By keeping locally data of the users on their device, FL improves privacy by design. However, FL can disclose sensitive information via model updates that are based on the training data. Indeed, any useful ML model reveals something about the population from which the training data was drawn. Indeed, a classifier model for instance may reveal the features that characterize a given class or help construct data points that belong to this class. The first privacy violation is property inference: identification of the features that characterize each class, making it possible to construct representatives of these classes through model inversion attacks [18]. Another privacy violation is attribute inference [38]: the leak of personal and unintended information (i.e., properties that hold for certain subsets of the training data, but not generically for all class members). The last privacy violation in our setting is membership inference [37]: given an exact data point, determine if it was used to train the model.

Memorization of training data by deep neural networks enables an adversary to conduct all these privacy violations. Firstly, this memorization usually combined with overfitting of the model are exploited by an adversary to conduct a membership inference attack in order to discriminate if a user has been part of the training or not [32]. This attack has a serious privacy implication especially if the learning model is related to sensitive information (e.g., presence of a certain pathology). Secondly, as deep-learning models come up with separate internal representations of all kinds of features, some of which are unpredictable and independent of the task being learned, the memorization of the training data can be leveraged by an adversary to infer a sensitive attribute [29]. In addition, due to the distributed nature of FL, passive and active inference attacks can be conducted by any participant or by the server.

Introducing Differential Privacy in the Stochastic Gradient Descent (DP-SGD) [1, 31] has been proposed to reduce the inference capability of an adversary, however this solution significantly damages the accuracy of the model and its capacity to converge [42, 46]. In addition, the noise calibration and the management of the privacy budget is not trivial. Other defenses propose to reduce the overfitting [34] but inherently decrease the utility. Secure aggregation relying on a cryptographic scheme has been also proposed [6, 8, 9]. Although the overhead of the underlying cryptographic schemes tends to be reduced, the management of this overhead and the participation of the server raises questions.

2.4 Mixnets

The concept of mixing information to make them indistinguishable or unlinkable is not new. Mix networks (Mixnets) [12] uses this concept to provide a proxy-based anonymity system. This system aims to provide unlinkability between the message sent by an user,

and the message received by the destination. More precisely, to prevent traffic analysis attacks, Mixnets route each message of the user through a set of anonymity servers called mixes. Mixes collect and shuffle (or mix) many messages before to route then to the destination. A variety of mixnets have been proposed including Aqua [26], Riffle [25], and Mixminion [17] addressing differently the tradeoff between anonymity, latency and bandwidth.

The limitation of these systems are similar to Tor, it is difficult for a user to determine which edge is uncompromised and powerful adversaries controlling both ends of the circuit can still deanonymize clients.

2.5 Intel SGX

The MixNN proxy relies on a Trusted Execution Environment (TEE), which leverages custom microprocessor zones, to enforce isolation, confidentiality and integrity of code and data. Specifically, we use Intel Software Guard Extensions (SGX) [3, 15] which defines the concept of enclave. The memory of an enclave is encrypted and cannot be directly accessed by other system software even by privileged code (e.g., the operating system or hypervisor). Enclaves can be attested to prove that the code running in the enclave is the one intended, and that it is running on a genuine Intel SGX platform. Once attested, enclaves can be provisioned with secret data by using authenticated secure channels. Moreover, enclaves can persist secret data outside the trusted zone by using a sealing mechanism. However, such protection comes with resource constraints. More precisely, only 96 MB out of the 128 reserved for the enclave can be used by applications. Although virtual and dynamic memory support is available [13, 14, 27], it incurs significant overheads in paging (i.e., the sealing and unsealing operations used an encryption key derived from the CPU hardware). However, [30] evaluates the usage of TEEs on mobile devices and reports a small system overheads at the client-side.

3 SYSTEM AND ADVERSARY MODEL

Before presenting MixNN, we describe our assumptions and the considered threat model. The operating flow of MixNN involves three premises with different level of trust, namely: (i) the client machine; (ii) the MixNN proxy; and (iii) the aggregation server.

First, we assume that the client machine is trusted. This includes the training data and all the computations performed locally. We do not consider malicious users trying to poison the model or to introduce backdoors.

Second, we assume that the MixNN proxy is running inside an Intel SGX enclave on an untrusted node. An adversary is thus not able to compromise the behavior or the data of the proxy. In addition, the remote attestation provided by Intel allows any participants to control that the MixNN proxy conforms to the expected behavior and has not been tampered with, and that it is running securely within an enclave on an Intel SGX enabled platform. However, an adversary can monitor the node where the MixNN proxy is deployed, possibly physically (e.g., monitoring network traffic, power consumption or memory access patterns). Consequently, an adversary can leverage side channel attacks [11] to infer information. We assume that the SGX enclave has generated a public and private key pair (k_{pub} and k_{priv}). As this keys generation is part of the proxy behavior and can be verifiable by the remote attestation, we do not assume collusion between the enclave and any party (e.g., the aggregation server).

Lastly, we consider an honest but curious aggregation server. This server builds a model for a main classification task through a federated learning scheme but also aims to infer membership and sensitive attributes from participants. This aggregation server conducts passive attacks. Specifically, it passively follows the FL operational flow and exploits auxiliary knowledge to infer sensitive information from participants. We consider an adversary with two types of auxiliary knowledge. First, we consider an adversary able to collect or to use a public dataset with similar raw data and statistical properties (including the sensitive attribute). This auxiliary knowledge allows the adversary to train a model to infer the sensitive attribute from the model updates sent to the aggregation server by each participant. Second, we also consider an adversary able to collect raw data from each participant. This second auxiliary knowledge, in turn, is used by the adversary to evaluate each received model update through the data of each participant in order to link a model update to a specific participant. More details about the inference attacks are described in Section 5.2. Finally, we consider protected exchanges between participants and the MixNN proxy (i.e., parameter updates are encrypted by using the public k_{pub} of the SGX enclave).

4 MIXNN FRAMEWORK

In this section, we first present an overview of the MixNN (Section 4.1), the equivalence in terms of utility with a classical FL (Section 4.2), and then give implementation details (Section 4.3).

4.1 Overview

To avoid inference attacks during the learning process of a service using a federated learning scheme, MixNN operates as depicted in Figure 3. To use MixNN, users have only to configure its system to use a proxy for the associated traffic (e.g., through the configuration of its browser). As such, users seamlessly get protected without changing their habits. More precisely, compared to the classical FL pipeline (Figure 2), all parameter updates will be sent to the MixNN proxy instead of the aggregation server. To secure these updates, they are encrypted with the public key of the enclave (i.e., k_{pub}) to ensure that only the MixNN proxy is able to read and process them. Once loaded in the enclave, the proxy decrypts and stores the parameter updates of each layer in different lists. The proxy then pick at random one update for each layer in the associated list to generate the message containing the parameter updates to send to the aggregation server. Note that the proxy needs to initialize first each list with k updates before to send updates to the aggregation server.

The rest of the workflow remains unchanged compared to the classical one. The server aggregates the parameter updates to generate a global model which will be disseminated to all participants. Participants will then refine this model locally with their personal data before sending the parameter updates to the MixNN proxy.

The accuracy of the global model remains unchanged with or without using MixNN. Indeed, whether mixed or not, the aggregation of parameter updates of each layer are identical. In contrast,

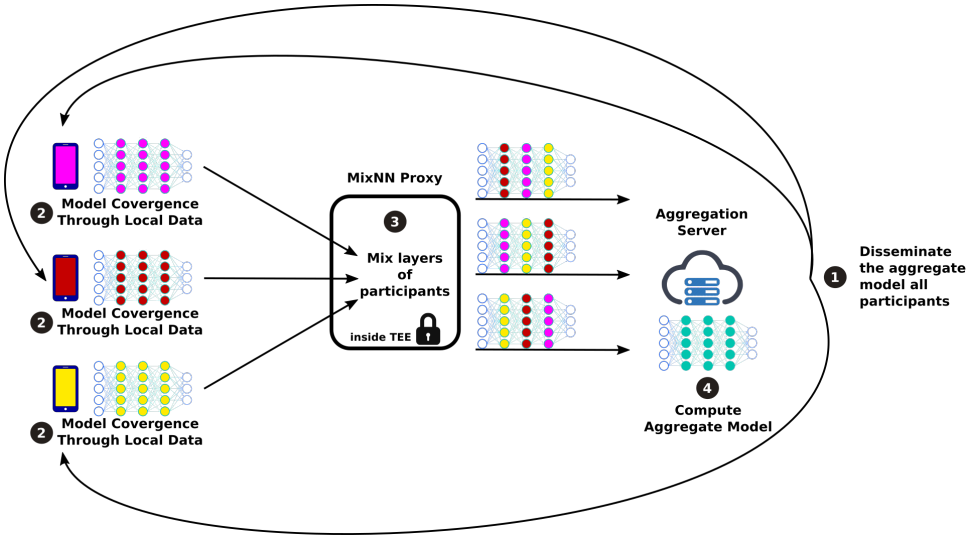


Figure 3: MixNN introduces a proxy which receives the parameter updates from each participant, shuffle them to remove attribute footprint before to route them to the aggregation server.

the privacy leakage through the footprint of parameter updates returned by participants is drastically reduced. Specifically, by receiving an update mixing information from different users (breaking potential footprints), the aggregation server is not able to infer any sensitive information. Consequently, MixNN is able to drastically improve privacy without compromising the accuracy of the system (i.e., no trade-off between utility and privacy). Indeed, by design, MixNN provides the same utility than a classical FL scheme. It is worth mentioning that the behavior of the proxy can adopt another strategy. For instance, the proxy can aggregate itself batch of model updates through averaging scheme or more robust estimators against model poisoning or backdoor attacks [5] such as geometric median. In this case, the utility-privacy-performance trade-off can change.

4.2 Utility Equivalence

By design, MixNN provides the same utility than a classical FL scheme. In this section, we prove this equivalence.

Let C be the number of participants sending their updates to the proxy. We show in this section that whether the participants use MixNN or not, the resulting aggregated model is the same. We assume that the considered MixNN proxy has enough information to send L updates to the server. Then the proxy creates a sequence (M_{ij}) such that $\forall (j_1, j_2) \in \{1, \cdots, n\}^2$ with $j_1 \neq j_2$ and $\forall i \in \{1, \cdots, L\} \quad M_{ij_1} \neq M_{ij_2}$. And also such that $\forall (i_1, i_2) \in \{1, \cdots, L\}^2$ with $i_1 \neq i_2$ and $\forall j \in \{1, \cdots, n\} \quad M_{i_1 j} \neq M_{i_2 j}$.

According to previously defined notation for the nodes of a neural network, we define the t-th layer of the c-th participant of the proxy by $\left(\theta_{.}^{.t}\right)^c$. In the following matrix, each line is a model sent by the proxy. We remark that each combination of participant/layer appears once and only once in the matrix. This is a fundamental assumption regarding the equality of accuracy level between traditional FL and MixNN .

$$A = \begin{pmatrix} \left(\theta_{.}^{.1}\right)^{M_{11}} & \left(\theta_{.}^{.2}\right)^{M_{12}} & \cdots & \left(\theta_{.}^{.n}\right)^{M_{1n}} \\ \left(\theta_{.}^{.1}\right)^{M_{21}} & \left(\theta_{.}^{.2}\right)^{M_{22}} & \cdots & \left(\theta_{.}^{.n}\right)^{M_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \left(\theta_{.}^{.1}\right)^{M_{L1}} & \left(\theta_{.}^{.2}\right)^{M_{L2}} & \cdots & \left(\theta_{.}^{.n}\right)^{M_{Ln}} \end{pmatrix}$$

Now, with the regular FL procedure, the information sent by the participant is:

$$B = \begin{pmatrix} \left(\theta_{.}^{.1}\right)^{1} & \left(\theta_{.}^{.2}\right)^{1} & \cdots & \left(\theta_{.}^{.n}\right)^{1} \\ \left(\theta_{.}^{.1}\right)^{2} & \left(\theta_{.}^{.2}\right)^{2} & \cdots & \left(\theta_{.}^{.n}\right)^{2} \\ \vdots & \vdots & \ddots & \vdots \\ \left(\theta_{.}^{.1}\right)^{C} & \left(\theta_{.}^{.2}\right)^{C} & \cdots & \left(\theta_{.}^{.n}\right)^{C} \end{pmatrix}$$

We note $Agr : \mathcal{M}(C \times n) \longrightarrow \mathcal{M}(1 \times n)$ the aggregation function which makes the mean of the columns. We show that $Agr(A) = Agr(B)$.

$$Agr(A) = \left(\frac{1}{L} \sum_{i=1}^{L} \left(\theta_{.}^{.1}\right)^{M_{i1}}, \frac{1}{L} \sum_{i=1}^{L} \left(\theta_{.}^{.2}\right)^{M_{i2}}, \cdots, \frac{1}{L} \sum_{i=1}^{L} \left(\theta_{.}^{.n}\right)^{M_{in}} \right)$$

$$Agr(B) = \left(\frac{1}{C} \sum_{c=1}^{C} \left(\theta_{.}^{.1}\right)^{c}, \frac{1}{C} \left(\theta_{.}^{.2}\right)^{c}, \cdots, \frac{1}{C} \sum_{c=1}^{C} \left(\theta_{.}^{.n}\right)^{c} \right)$$

We make the additional assumption that $L = C$ which means that the MixNN proxy waits for the C participants to send their updates before mixing. Which gives us that:

$$Agr(A) = Agr(B) \iff$$

$$\left[\forall l \in \{1, \cdots, L\} \quad \sum_{c=1}^{C} \left(\theta_{.}^{.l}\right)^{M_{cl}} = \sum_{c=1}^{C} \left(\theta_{.}^{.l}\right)^{c} \right]$$

Which is true since our assumption on (M_{ij}) gives us that $\varphi : \{1, \cdots, C\} \longrightarrow \{1, \cdots, C\} \quad c \mapsto M_{cl}$ is a bijective mapping.

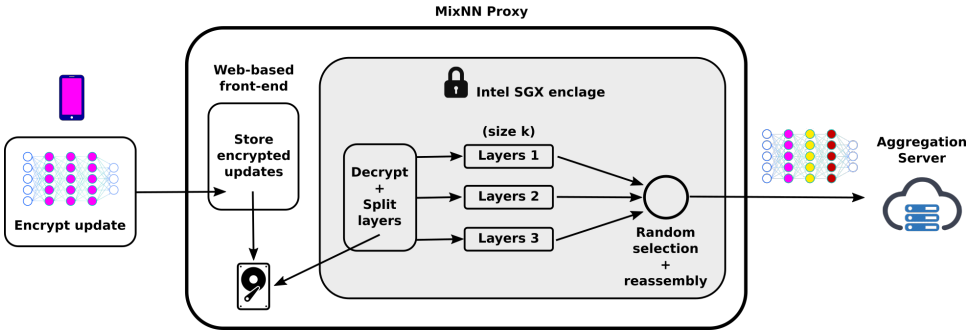


Figure 4: Implementation and data-flow of the MixNN proxy.

4.3 Implementation

MixNN is implemented inside an Intel SGX enclave to protect its behaviors and confidentiality even if it is deployed on an untrusted node. In our implementation, the SGX enclave first generates an RSA key pair including a private and a public key of 4,096 bits, and parameters update is encrypted by participants using the public key of the enclave with AES-256-GCM algorithm, ensuring only the enclave could access its content. The data-flow of the MixNN proxy is depicted Figure 4. A web-server is used as a front-end of our proxy, receiving parameter updates from participants and saving them on disk. To improve performance, these updates are sent in a binary format. The SGX enclave then scans the filesystem for new updates. Each new update is decrypted and splited by cutting the binary block into pieces corresponding to the size of each layer. The parameters associated to each layer are then stored in different lists. The size of these lists (noted k) and the memory allocation according to the considered neural network models are initialized at the creation of the enclave. Once k parameter updates are received, those lists are full and each subsequent update will generate a mixed update, picking one random layer from each list. This generated update can then be forwarded to the aggregating server. Also our implementation takes advantage of the multi-threaded capabilities of Intel SGX with each thread processing one incoming parameter update and generating a mixed update if necessary.

To avoid side-channel attacks against SGX [11], the cost (i.e., the execution time) to process an update is constantly the same. Depending on the considered model, the size of a model can be important and not fit into the memory limit of the enclave (96MB), requiring encrypted storage outside the enclave. To avoid side-channel attack based on memory access, ORAM mechanisms (e.g., ZeroTrace [35]) can be adopted to carry out secure and oblivious access of data. The associated overhead is negligible in our context where updates are sent only periodically.

5 EVALUATION SETUP

In this section we presents the experimental setup used to evaluate MixNN, which includes datasets (Section 5.1), metrics (Section 5.2), baselines we compared against (Section 5.3), and the considered methodology (Section 5.4).

5.1 Dataset

We used two image recognition benchmark datasets (CIFAR10 and LFW) and two motion datasets for activity recognition (MotionSense and MobiAct) to assess MixNN.

CIFAR10 is a major image classification benchmarking dataset where the data records are composed of 60,000 32×32 RGB images where each record is mapped to one of 10 classes of common objects such as airplane, bird, cat, dog. There are 50,000 training images and 10,000 test images. The main task is the classification of the images. We artificially define 20 participants split into three groups with different preferences. We define 3 types of preference which corresponds to specific and non overlapping categories of images. The dataset is slightly balanced, two groups gather 6 participants and the last one gathers 8 participants. The profile of the participant is composed of 80% of images corresponding to its preferred classes, and the remaining 20% is composed of random images from other classes. The sensitive attribute is the preferences of the user.

MotionSense [33] contains data captured from an accelerometer (i.e., acceleration and gravity) and gyroscope at a constant frequency of 50Hz collected with an iPhone 6s kept in the front pocket. Overall, a total of 24 participants have performed six activities (i.e., going downstairs, going upstairs, walking, jogging, sitting and standing) during 15 trials in the same environment and conditions. The main classification task is the activity detection and the sensitive attribute is the gender of the users.

MobiAct [44] records the motion data from 58 subjects during more than 2,500 trials, all captured with a Samsung Galaxy S3 in the front pocket. This dataset includes signals recorded from the accelerometer and gyroscope at 20Hz. We only used the trials corresponding to the same activities as MotionSense in order to do the evaluation with the same settings. Similar to MotionSense dataset, the main classification task is the activity detection and the sensitive attribute is the gender of the users.

Labeled Faces in the Wild (LFW) [22] contains face images for face recognition with 13,233 total samples with images for 5,749 people. The dataset additionally has attributes such as age, race, gender, smile, facial hair, glasses etc. The main classification task is smile detection and the sensitive attribute is the gender of the users.

For CIFAR10, MotionSense and MobiAct datasets, we use a neural network composed of two convolutional layers and three fully connected layers for the classification task. For LFW, in turn, we use

a more complex architecture provided by Facebook, named Deep Face [40]. This neural network is composed of multiple convolutional, locally connected, maxpooling, and fully connected layers.

5.2 Evaluation Metrics

We evaluate MixNN through three complementary dimensions: utility, privacy and system performance.

To evaluate the utility of the target model, we consider the classification accuracy for the main task (e.g., the activity detection), noted *Model Accuracy*, measuring the ratio of number of correct predictions over the total number of predictions made.

To assess the privacy, we implemented both a membership and an attribute inference attack. First, we revisit the membership inference attack to compute an upper-bound of the risk of linkability between a model update and a participant. We consider here an adversary with an auxiliary knowledge about each participant (i.e., some raw data). The adversary (i.e., the aggregation server) is thus able to evaluate a received model update with the auxiliary data of each participant in order to link this (anonymous) model update to a specific participant. Specifically, the adversary predicts the link between a participant and the participant for which its auxiliary data produced the lowest loss. We report the *Linkability Accuracy* measuring the precision of this linkability. As an update produced by MixNN is composed of layers from different participants, we report an upper-bound by counting a good prediction if the most sensitive participant (i.e., the layer with the lowest loss) included in the update is linked with the correct participant.

Second, for the attribute inference attack, we train a random forest classifier to infer the sensitive attribute from the parameter updates. This classifier leverages auxiliary knowledge, specifically the raw data and the sensitive attribute of few participants (i.e., 3 males and 3 females in our case). To increase the number of model updates used for the training of this random forest classifier, each received model update is refined with each auxiliary data. More precisely, a model update received at one learning round by the adversary (i.e., the aggregation server in our case) is refined with the raw data of each participant part of the auxiliary knowledge and labeled with the same sensitive attribute. We report the success of the attribute inference, noted $Inference\ Accuracy$. This value indicates a data leakage according to the number of classes. For instance, with a balanced dataset over the gender, an accuracy different than 50% indicates that the adversary is able to identify the gender of a participant with an accuracy higher than random guess.

To evaluate the behavior of MixNN from a systems perspective, we consider the end-to-end latency which is the time spent by the proxy to route the parameter updates to the aggregation server.

5.3 Baselines

We compare the utility and privacy provided by MixNN against different comparative approaches. Firstly, we consider an approach using noisy gradient widely used in Differential Privacy studies [31, 49]. We use an implementation based on an introduction of Gaussian noise to the updates computed through a classical local training such as using in local differential privacy [31]. Secondly, we consider an model compression scheme using Pruning. Pruning the network results in reducing the format of the parameters of the neural network (e.g., from 32 bits to 16 bits). A compressed format reduces the overall memory bandwidth [20]. Lastly, we also consider a classical Federated Learning scheme.

5.4 Methodology

The dataset is split between training and testing, with 5/6 of trials used for training and validation and 1/6 for testing. For CIFAR10, the federated learning model is trained on 3 local epochs for a size of data batch of 32 samples on each learning rounds, the server aggregates 20 users on each of the 40 learning rounds. For MotionSense (and MobiAct), the training is (respectively) done on 2 (and 3) local epochs for batches of 256 samples for each of the 40 learning rounds, and the server aggregates 24 users for MotionSense and 58 users for MobiAct. For LFW, the training is done on 2 local epochs for batches of 8 samples for each of the 40 learning rounds, and the server aggregates 62 users. For every datasets, we use the "Adam" optimizer proposed by Tensorflow. We use 3-fold cross-validation in which the testing set is randomly generated from 1/3 of the users. Reported results correspond to average over 3 repetitions of each experiment. The experiments have been computed on Grid5000 []. The noise introduced consists on adding a Gaussian noise $\mathcal{N}(0, 0.1)$ on each scalars of the neural network weights. Finally, MixNN has been implemented within the SGX enclave of an Intel i5-8500 CPU.

6 EVALUATION

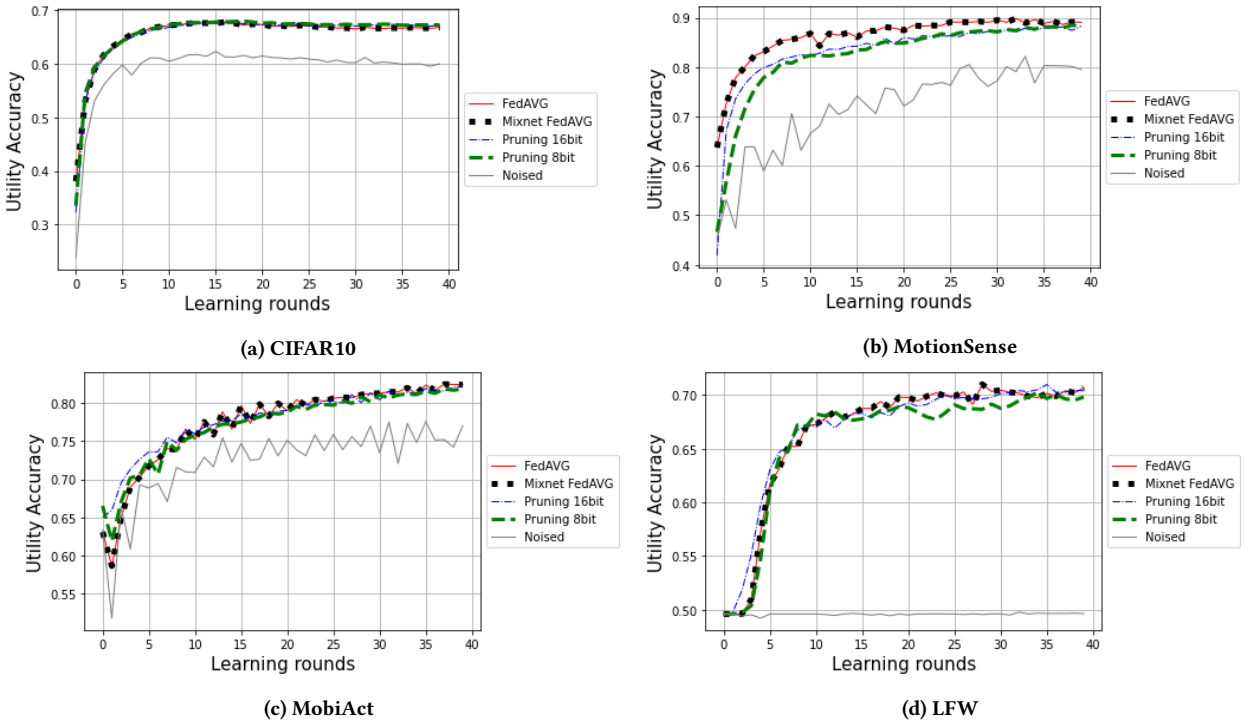
We now report the results in terms of utility (Section 6.1) and privacy (Section 6.2) provided by MixNN under the considered experimental setup (Section 5). We also analyse the robustness of MixNN against an aggregation server trying to defeat the protection. (Section 6.3) as well as its performance from a systems perspective (Section 6.4).

Our results show that MixNN efficiently reduces the information leakage through both a membership and an attribute inference attack without compromise on the accuracy of the model. We also show that MixNN introduces a negligible end-to-end latency.

6.1 No compromise with utility

In this section, we evaluate the capacity of MixNN to protect privacy without compromising the utility. We compare the accuracy performance for the main classification task provided by MixNN against a classic FL scheme (i.e., without MixNN proxy), a baseline using noisy gradient such as using in local differential privacy and a pruning strategy. Figure 5 reports the accuracy according to the learning round for all datasets. First, the results show that the same level of accuracy is provided by a standard FL scheme and MixNN with an accuracy growing according to the learning rounds. This result is expected due to the aggregation equivalence of both approaches. Second, the results show that noisy gradient provides 10% lower accuracy on average and slows down the convergence. Noisy gradient even breaks the learning capability in the case of the LFM dataset. Other approaches based on pruning slightly deteriorate the model's accuracy.

In practice, node churn (i.e., some participants do not participate to all learning rounds) results by the reception of fewer number of updates than expected. As the MixNN proxy waits the reception of k updates before to prepare the mixed updates (Section 4.3), lists



(a) CIFAR10 **(b) MotionSense**

(c) MobiAct **(d) LFW**

Figure 5: MixNN provides the same utility than a standard FL scheme, noisy gradient however decreases significantly the utility and slows down the convergence.

of updates (splited by layer) maintained in the MixNN proxy can gather information from different learning rounds, and might slow the convergence of the learning. However, this effect has only a limited impact (e.g., with 20% of updates missing at each round, a slowdown in convergence was observed only on the LFW dataset).

6.2 Prevent information leakage

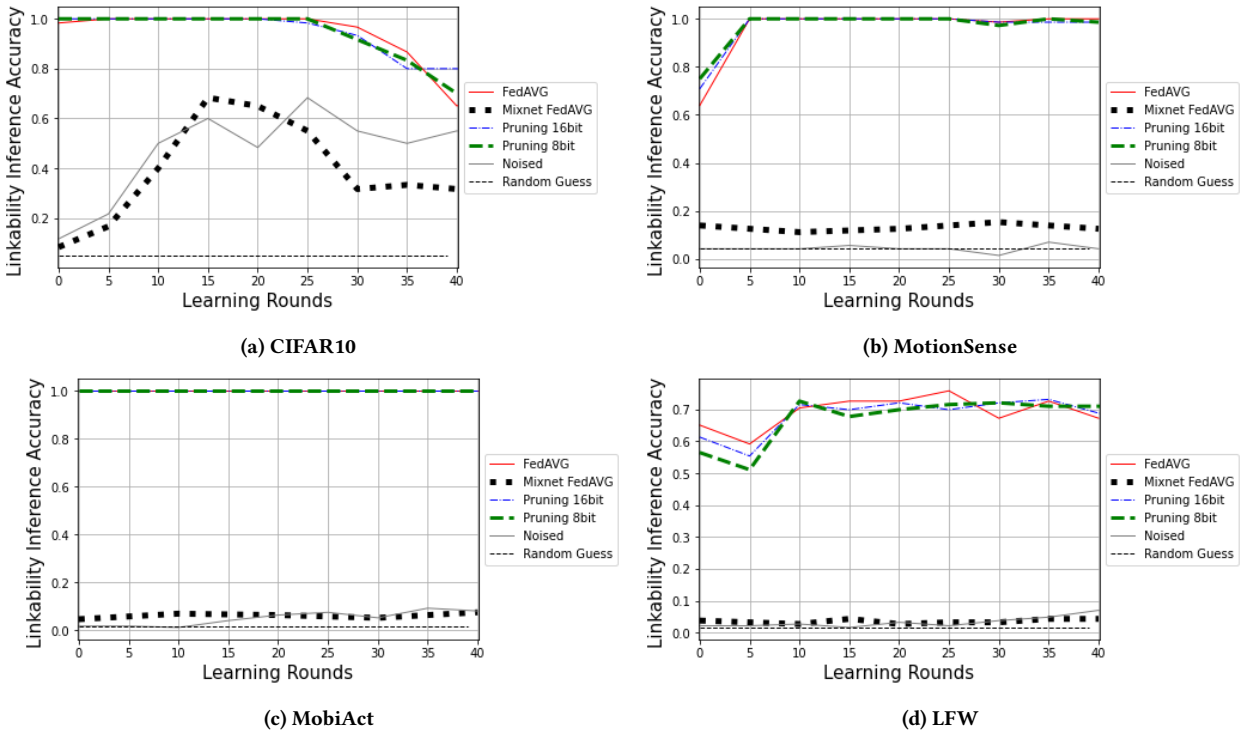
In this section, we evaluate the privacy leakage through both a membership and a sensitive attribute inference attacks.

6.2.1 Membership inference attack. For the membership inference attack, Figure 6 reports for all datasets the accuracy of the linkability between an update and a participant for MixNN, a classical FL, a pruning strategy using 16 and 8 bits and noisy updates according to a growing number of learning rounds. In all datasets, the pruning approaches and the classical federated learning provide roughly the same poor results: the updates are successfully linked to the correct participant (i.e., around 100% of accuracy), except for LFW with an accuracy around 70%. This means that the gradient vector returned by participants can be exploited to provide an efficient footprint to re-identify the associated participant. Our approach MixNN and the noisy gradient provide a near perfect protection (i.e., close to a random guess) for all datasets except for Cifar 10 which slightly reduce the linkability accuracy.

6.2.2 Attribute Inference Attack. For the attribute inference attack, the Figure 7 reports for all datasets the accuracy of the inference for MixNN, a classical FL, prunned updates at 16 and 8 bits and noisy updates according to a growing number of learning rounds. This attribute inference attack trends to provide better accuracy on fully converged models, so we evaluate the attack between the rounds 80 and 100 of the federated learning process. For all datasets, the accuracy of the attack for MixNN is closer to the random guess (i.e., the attacker is unable to learn the sensitive attribute) compared to other baselines approaches. The other comparative baselines, in contrast, fail to protect the sensitive attribute inference (e.g., the classical federated learning update leaks up to 52% of the clients gender in comparison to the random guess). For Cifar10, MixNN provides the best protection and fits to the random guess. The attack provides poor results, the reason is that the clients have artificial data distribution: there is model weights that can be optimal on every clients dataset. The retrain on the auxiliary data is then useless and the random forest classifier fails to learn such pattern on the sensitive attribute.

6.3 Robustness of the protection

MixNN shuffles model updates sent by participants. A malicious aggregation server could then try to break the protection by enumerating through possible combinations of the shuffled updates to reconstruct back the original update. We evaluate a worst case scenario where the aggregation server holds clients data. To rebuild a specific clients update, the server evaluates each mixed layers of updates by simply replacing this layer in the aggregated model. Given a dataset, the server builds back the model that give the lowest loss on it, layer by layer. We evaluate then how accurate the server



(a) CIFAR10 **(b) MotionSense**

(c) MobiAct **(d) LFW**

Figure 6: MixNN better prevents the membership attack compared to a classical FL and a pruning strategy.

was in this rebuilding process. This process fails in more than 80% of the clients across all datasets and this process is very costly in computation time. Breaking the mixing strategy of MixNN seems to be difficult to achieve.

6.4 System performance

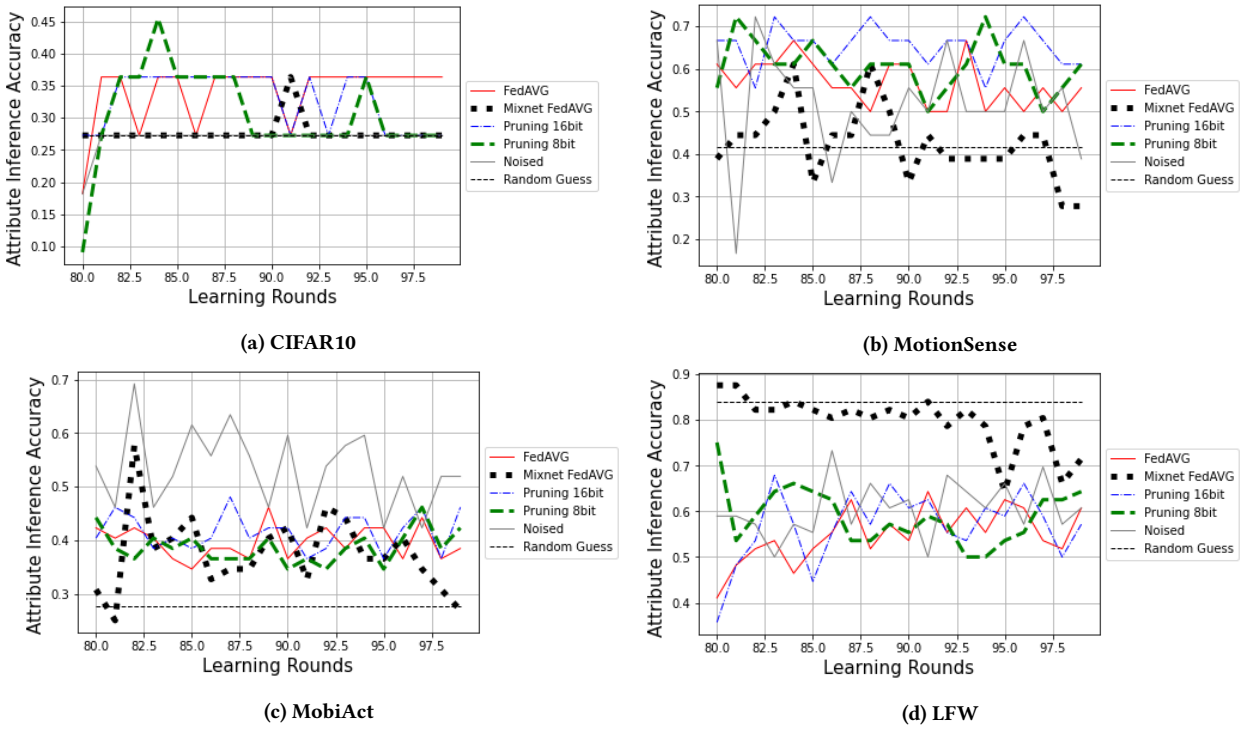
We implemented MixNN proxy inside an SGX Enclave to evaluate its system performance. As described Section 4.3, this MixNN proxy receives the update parameters as an encrypted binary file, decrypts inside the enclave each layer and stores them in the trusted memory of the enclave before to prepare new updates with mixed layers and sends them to the untrusted aggregation server. Figure 9 depicts the time spent by the MixNN proxy to process a growing number of updates. We compare the scalability of MixNN against three comparison baselines. Specifically, these comparison baselines include a federated average aggregation (i.e., the enclave receives and decrypts the updates and then does an average of all received updates and send all updates with the averaged values to the untrusted server), a decryption only proxy (i.e., the enclave decrypts the received updates and forwards them to the aggregation and untrusted server), and a simple proxy (i.e., the proxy sends every received update to the aggregation and untrusted server without passing them to the enclave). Results show that time spent by MixNN to process the updates is linear according to the number of updates. As the learning round of classical FL scheme is usually not conducted at high rate (e.g., only when the device is plugged in and has a WiFi connection), this short delay introduced by MixNN is negligible. In addition, the constant processing time over all updates for a given model (i.e., the MixNN proxy waits to receive a constant number of model updates before mixing them) reduces the surface for side channel attacks.

This experiment uses a model designed for activity recognition (i.e., MotionSense dataset), with two convolutional layers and four fully connected layers. Each update of this model uses 3.3MB inside the enclave, 4.4MB while encrypted.

7 RELATED WORK

The incentive behind using FL is to collectively build a learning model with better accuracy than if each user trained a model with their own data. The goal is to improve the accuracy as much as possible but several dimensions have an influence. The standard FL scheme [28] learns one global model and replicates it locally on every client. However heterogeneity of data across user devices can severely degrade performance of standard federated averaging for ML learning applications, especially for atypical users. Indeed, one unique model cannot cope with the heterogeneity of data and provide the best utility for all users [10]. To address this data heterogeneity, several approaches have been proposed such as local adaptation [4, 46] and clustering [36]. Specifically, the clustering mechanism proposed in [36] also leverages a similarity metrics between the model updates sent back by participants (similar to MixNN) to cluster the population.

[32] designs passive and active inference algorithms for federated learning. However, this work only targets membership inference. In



(a) CIFAR10

(b) MotionSense

(c) MobiAct

(d) LFW

Figure 7: MixNN better prevents sensitive attribute leakage compared to using noisy gradient.

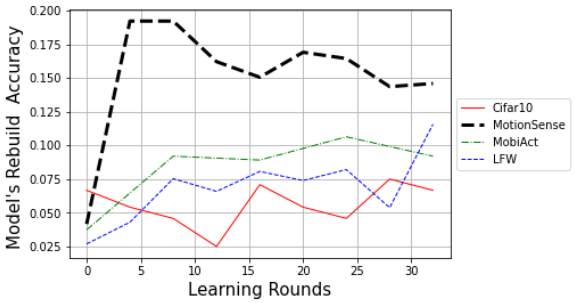


Figure 8: Reconstruction of the model updates is costly and gives a poor accuracy.

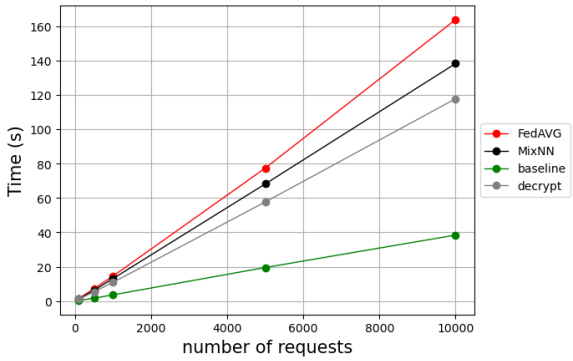


Figure 9: Time spent by MixNN to process the updates is linear according to the number of updates.

addition, while this attack also exploits the privacy vulnerabilities of the SGD algorithm, authors used a neural network to classify if a participant is a member of the training data or not a member. [49] also exploits the gradient exchange to infer private training data of participants. To do that, authors iteratively optimize "dummy" inputs and labels to minimize the distance between dummy gradients and real gradients. Once the optimization finished, the dummy data is close to the private training data. [23], in turn, investigates the guarantees of differentially private SGD but via data poisoning attacks.

Running MixNN in an Intel SGX enclave improves trust and confidentiality through an isolated execution environment. However, this TEE is still vulnerable to side channel attacks [11, 43]. The most common countermeasure is to use data oblivious algorithms. The objective of this technique is to eliminate the link between the nature of data inputs and the execution of the program (e.g., through the execution time or memory footprints). To achieve that, the obfuscation technique consists to hide potential patterns by making them all uniform regardless of the considered data. To reduce its inherent cost, the considered data oblivious algorithm needs to be chosen carefully according to the application [2].

8 CONCLUSION

We presented MixNN, a proxy-based privacy-preserving framework to prevent both membership and sensitive attribute inference attacks conducted from a curious aggregation server exploiting the model updates. MixNN breaks the footprint leaked in the model updates by mixing layers between multiple participants. As this mixing strategy does not impact the result of the model aggregation performed by the server, the privacy improvement of MixNN does not compromise the utility of the model learned collaboratively. We experimentally evaluated MixNN with different benchmark datasets and compared it against a state-of-the-art baselines using local differential privacy, a pruning strategy and a classical FL. Results show MixNN provides the same model accuracy than a classical FL scheme (i.e., the same utility) while providing a better protection (i.e., a better privacy) compared to other baseline approaches.

ACKNOWLEDGMENTS

This work has been partly supported by ANR grant ANR-20-CE23-0013 and by the Face Foundation under the Trusty-AI project.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *CCS*.

[2] A K M Mubashwir Alam, Sagar Sharma, and Keke Chen. 2020. SGX-MR: Regulating Dataflows for Protecting Access Patterns of Data-Intensive SGX Applications. *arXiv:2009.03518* (2020).

[3] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative technology for CPU based attestation and sealing. In *HASP*.

[4] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated Learning with Personalization Layers. *arXiv:1912.00818* (2019).

[5] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[6] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *CCS*. 1253–1269.

[7] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *MLSys* (2019).

[8] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2016. Practical secure aggregation for federated learning on user-held data. *arXiv:1611.04482* (2016).

[9] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*. 1175–1191.

[10] Antoine Boutet, Carole Frindel, Sébastien Gambs, Théo Jourdan, and Claude Rosin Ngueveu. 2021. DYSAN: Dynamically sanitizing motion sensor data against sensitive inferences through adversarial networks. In *AsiaCCS*.

[11] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. 2017. Software Grand Exposure: SGX Cache Attacks Are Practical. In *WOOT*.

[12] David L. Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun.* 24, 2 (Feb. 1981), 84–90.

[13] Feng Chen, Chenghong Wang, Wenrui Dai, Xiaoqian Jiang, Noman Mohammed, Md Momin Al Aziz, Md Nazmus Sadat, Cenk Sahinalp, Kristin Lauter, and Shuang Wang. 2017. PRESAGE: Privacy-preserving genetic testing via software guard extension. *BMC medical genomics* 10, 2 (2017), 48.

[14] Feng Chen, Shuang Wang, Xiaoqian Jiang, Sijie Ding, Yao Lu, Jihoon Kim, S Cenk Sahinalp, Chisato Shimizu, Jane C Burns, Victoria J Wright, et al. 2016. Princess: Privacy-protecting rare disease international network collaboration via encryption through software guard extensions. *Bioinformatics* 33, 6 (2016), 871–878.

[15] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. *IACR Cryptology ePrint Archive* 2016, 086 (2016), 1–118.

[16] Emiliano De Cristofaro. 2020. An Overview of Privacy in Machine Learning. *arXiv:2005.08679* (2020).

[17] G. Danezis, R. Dingledine, and N. Mathewson. 2003. Mixminion: design of a type III anonymous remailer protocol. In *2003 Symposium on Security and Privacy, 2003.* 2–15.

[18] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *CCS*. 1322–1333.

[19] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. 2018. Property Inference Attacks on Fully Connected Neural Networks Using Permutation Invariant Representations. In *CCS*. 619–633.

[20] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. 2016. EIE: Efficient Inference Engine on Compressed Deep Neural Network. In *ISCA*. 243–254.

[21] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[22] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Technical Report 07-49. University of Massachusetts, Amherst.

[23] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. 2020. Auditing Differentially Private Machine Learning: How Private is Private SGD? *arXiv:2006.07709* (2020).

[24] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. 2020. Revisiting Membership Inference Under Realistic Assumptions. *arXiv:2005.10881* (2020).

[25] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. 2016. Riffle: An Efficient Communication System With Strong Anonymity. *Proc. Priv. Enhancing Technol.* 2016, 2 (2016), 115–134.

[26] Stevens Le Blond, David Choffnes, Wenxuan Zhou, Peter Druschel, Hitesh Ballani, and Paul Francis. 2013. Towards Efficient Traffic-Analysis Resistant Anonymity Networks. *SIGCOMM Comput. Commun. Rev.* 43, 4 (Aug. 2013), 303–314.

[27] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. 2016. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *HASP*.

[28] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv:1602.05629* (2017).

[29] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2018. Exploiting Unintended Feature Leakage in Collaborative Learning. *arXiv:1805.04049* (2018).

[30] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. 2021. PPFL: Privacy-Preserving Federated Learning with Trusted Execution Environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (Virtual Event, Wisconsin) *(MobiSys '21)*. Association for Computing Machinery, New York, NY, USA, 94–108. https://doi.org/10.1145/3458864.3466628

[31] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. 2021. Toward Robustness and Privacy in Federated Learning: Experimenting with Local and Central Differential Privacy. *arXiv:2009.03561* (2021).

[32] 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 *S&P*.

[33] J. L. Reyes-Ortiz. 2015. *Smartphone-based human activity recognition.* Springer.

[34] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. *arXiv:1806.01246* (2018).

[35] Sajin Sasy, Sergey Gorbunov, and Christopher W. Fletcher. 2018. ZeroTrace : Oblivious Memory Primitives from Intel SGX. In *NDSS*.

[36] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2019. Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization under Privacy Constraints. *arXiv:1910.01991* (2019).

[37] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks against Machine Learning Models. *arXiv:1610.05820* (2017).

[38] Congzheng Song and Vitaly Shmatikov. 2020. Overlearning Reveals Sensitive Attributes. *arXiv:1905.11742* (2020).

[39] Liwei Song and Prateek Mittal. 2020. Systematic Evaluation of Privacy Risks of Machine Learning Models. *arXiv:2003.10595* (2020).

[40] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *CVPR*.

[41] Florian Tramèr and Dan Boneh. 2020. Differentially Private Learning Needs Better Features (or Much More Data). *arXiv:2011.11660* (2020).

[42] Laurens van der Maaten and Awni Hannun. 2020. The Trade-Offs of Private Prediction. *arXiv:2007.05089* (2020).

[43] Stephan van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom. 2020. SGAxe: How SGX Fails in Practice. https://sgaxeattack.com/.
[44] George Vavoulas, Charikleia Chatzaki, Thodoris Malliotakis, M. Pediaditis, and M. Tsiknakis. 2016. The MobiAct Dataset: Recognition of Activities of Daily Living using Smartphones. *ICT4AWE* (2016).
[45] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *CSF*. 268–282.
[46] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. 2020. Salvaging Federated Learning by Local Adaptation. *arXiv:2002.04758* (2020).
[47] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. 2020. Dataset-Level Attribute Leakage in Collaborative Learning. *arXiv:2006.07267* (2020).
[48] Benjamin Zi Hao Zhao, Aviral Agrawal, Catisha Coburn, Hassan Jameel Asghar, Raghav Bhaskar, Mohamed Ali Kaafar, Darren Webb, and Peter Dickinson. 2021. On the (In)Feasibility of Attribute Inference Attacks on Machine Learning Models. *arXiv:2103.07101* (2021).
[49] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep Leakage from Gradients. *arXiv:1906.08935* (2019).